Un modelo computacional para representar videojuegos multijugador de detectives e implementar el comportamiento de participantes automáticos

Alejandro Villarín Prieto

DOBLE GRADO EN INGENIERÍA INFORMÁTICA Y MATEMÁTICAS

FACULTAD DE INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

TRABAJO DE FIN DE GRADO

9 de septiembre de 2016

Director: Federico Peinado Gil
Codirector: Nahum Álvarez Ayerza

Agradecimientos

Agradecer encarecidamente a mi director Federico Peinado Gil y a mi codirector Nahum Álvarez Ayerza todos sus esfuerzos para realizar este trabajo, sin los cuales jamás hubiera acabado con éxito.

Índice general

Índice de figuras

Resumen

En la actualidad, los videojuegos no llegan a alcanzar el nivel de realismo esperado debido, entre otros factores, al trato que se hace del conocimiento de sus personajes. En muchas ocasiones estos personajes no son capaces de actualizar su conocimiento acerca de lo ocurrido convenientemente, dando lugar a situaciones extrañas.

Este trabajo aborda el problema de la gestión de conocimiento, el razonamiento y la comunicación entre personajes de videojuego controlados por computador, algo que la industria del entretenimiento interactivo está muy interesada en conseguir para aumentar la jugabilidad y la credibilidad de sus obras.

Para explorar las posibilidades de construir personajes con una inteligencia artificial capaz de extraer conocimiento sobre lo que perciben en su entorno, y propagarlo a otros de manera realista, imperfecta o incluso malintencionada, se propone un escenario ficticio propio de un videojuegos multijugador de detectives, donde además de participantes humanos, vamos a explorar la posibilidad de implementar participantes automáticos.

El escenario, de temática fantástica y humor, consiste básicamente en un laberinto de un laboratorio donde un grupo de ratones pueden moverse y realizar acciones por la noche. A la mañana siguiente el científico del laboratorio inspeccionará el estado del laberinto e intentará averiguar ratones han causado desperfectos para castigarlos.

En este trabajo explicamos el proceso llevado a cabo para modelar este juego y representar computacionalmente el conocimiento y los pasos de razonamiento que deben dar los participantes para jugar de forma plausible al mismo. Con este modelo sentamos las bases de un armazón que permita explorar diferentes estrategias de resolución de problemas ante toda una familia de escenarios posibles donde hay personajes que intercambian infor-

mación tratando de maximizar su puntuación en el juego.

Finalmente se ofrece una solución al problema con ratones que son conscientes de las acciones que pueden acarrear consecuencias que han realizado y por lo tanto mienten en sus declaraciones para evitar ser castigados. Los ratones inocentes pretenden sacar a la luz la verdad. El científico sigue un método de investigación que le lleva a dar con un ratón sospechoso.

Palabras clave: Conocimiento Impreciso, Inteligencia Artificial, Personajes Creíbles, Agentes Inteligentes, Gestión del Conocimiento, Razonamiento, Videojuegos.

Abstract

Nowadays, videogames do not achieve the expected level due to the treatment that is done to the knowledge of the characters. Usually, this characters cannot update their knowledge about what happened correctly, provoking strange situations.

This work takles problems related to knowledge management, reasoning and characters comunication in videogames, something that this industry is very interested in achieving in order to improve its gameplay and the credibility of its works.

In order to explore all the possibilities of creating characters whose artificial intelligence can get the knowledge of its enviroment and propagate it to others in a realistic, imperfect or even malicious way, we propose a ficticious stage of a multiplayer videogame about detectives, where moreover characters led by the player, we are going to explore the possibility of introducing non player characters.

The stage, which is about a fantastic and funny topic, is mainly a maze in a laboratory where a group of mice can move a carry out actions during the night. In the morning, the scientist of the laboratory will check the maze conditions and he will try to discover which mice made damage to punish them.

In this work we explain the process carried out to shape this game and the process to represent the knowledge and the reasoning steps that the characters must follow in order to play in a right way to the same one. This model sets the foundations of a framework that allow us to explore the different strategies to solve problems about possible stages where the characters exchange information and try to maximize their punctuation in the game.

Finally, we propose a solution to the problem that consist in a group of

mice conscious of the consequences of their actions so they decide to lie in their declarations to avoid be punished. The innocent mice try to show the truth while the scientist follows an research method to know who is the guilty mouse.

Keywords: Ambiguous Knoledge, Artificial Intelligence, Believable Characters, Intelligent Agents, Knowledge Management, Reasoning, Videogames.

Capítulo 1

Introducción

La narración interactiva es una disciplina que ha cobrado importancia en los últimos años, principalmente impulsada por la industria de los videojuegos, debido a la necesidad de ofrecer experiencias multimedia más atrayentes para los usuarios. Uno de los factores necesarios para mejorar dicha experiencia recae en la generación de personajes con un comportamiento creíble, un aspecto que tradicionalmente ha sido pasado por alto en los videojuegos, limitándose a generar comportamientos dirigidos por patrones, dado que en los distintos géneros de videojuegos no era necesaria una mayor sofisticación.

Sin embargo recientemente los juegos con un importante aspecto narrativo (ya sean aventuras gráficas tradicionales, o géneros híbridos) han cobrado cada vez más popularidad, generando la necesidad de personajes con una inteligencia más cercana a lo que nos parecería natural en un humano.

De hecho, los juegos de misterio donde hay que descubrir al culpable de un crimen consiguen captar la atención del público rápidamente. Sin embargo, en dichos juegos los personajes son incapaces de entender el escenario en el que se mueven. Actúan siempre siguiendo un modelo que en ocasiones pueden llevarlos a realizar acciones contraproducentes.

Un claro ejemplo de esto es el videojuego ***The Black Mirror*** (The Black Mirror, Future Games, 2003), el cual fue duramente criticado por ser demasiado lineal. Otro ejemplo más evidente, aunque de distinto género, son los primeros títulos de la famosa saga de videojuegos de plataformas ***Crash Bandicoot*** (Crash Bandicoot, Naughty Dog, 1996), donde el jugador tiene que pasar una serie de niveles en los que las trampas y los adversarios siguen siempre la misma ejecución.

Esta falta de conocimiento se amplifica debido a que los personajes no son capaces de sacar sus propias conclusiones de una situación, salvo que se hayan definido previamente para actuar como si estuviera deduciendo algo. Podemos ver esta situación en juegos de acción y otros que mezclan partes de acción con infiltración como ***Batman: Arkham Knight*** (Batman: Arkham Knight, Rocksteady Studios, 2015). Muchos de estos juegos carecen de una lógica a la hora de buscar intrusos, ya que no son capaces de deducir donde se encuentra el sospechoso que se esconde en cuanto sale de su visión directa.

Un caso similar es el de un vigilante omnisciente en su parcela y que carece totalmente de percepción y conocimiento fuera de ella, como ocurre en ***Sly Cooper*** (Sly Cooper and the Thievius Raccoonus, Sucker Punch Productions, 2002). Debido a esto, para que el jugador huya del vigilante que le ha visto debe salir fuera de la zona que tiene asignada dicho vigilante. El vigilante te perseguirá hasta la frontera de su zona, y una vez atravesada te dejará en paz, aunque siga teniendo contacto visual. Sin embargo, puede que esa zona sea tan amplia como para pensar que el guardia no debería encontrarte aunque todavía estés dentro.

Esto se debe a que, en general, los personajes de los videojuegos actúan siguiendo una máquina de estados que cuenta con dos modos principales: el primero es la búsqueda de posibles amenazas y el segundo es la reacción cuando se ha detectado la amenaza. Sin embargo, esto le resta mucha credibilidad al juego, ya que en una situación real entre ambos modos hay un amplio rango de posibles situaciones. Este problema está presente en muchos juegos recientes como ***Uncharted 4*** (Uncharted 4: A Thief's End, Naughty Dog, 2016) o ***Assassin's Creed III: Liberation*** (Assassin's Creed III: Liberation, Ubisoft, 2014).

La información del entorno no solo es útil para lo toma de mejores decisiones. El conocimiento que nuestros personajes adquieren durante la ejecución se convierte en una pieza fundamental si queremos mantener conversaciones que muestren una sensación de mayor autenticidad.

Toda la información que percibimos del entorno puede ser almacenada para posteriormente usarlo en conversaciones o como defensa frente a acusaciones. Este es un tema importante para darle realismo a los videojuegos, ya que se evita tener argumentos predefinidos dando mayor flexibilidad a la historia.

Es común que los videojuegos sigan un hilo argumental con pocas variaciones, evitando así tener que modificar las conversaciones entre los personajes. Sin embargo, hay ocasiones en los que el jugador se desvía del rumbo que se espera provocando grandes incongruencias entre las conversaciones que tienen lugar y lo que realmente ha ocurrido.

Un caso como este se da en el clásico juego de ***Pokémon*** (Pokémon Red Version y Blue Version, Game Freak, 1996). El juego consiste en recorrer las ciudades, las cuales normalmente estamos obligados a visitar siguiendo el orden que los desarrolladores eligieron. En esas ciudades debemos enfrentarnos a los líderes del gimnasio para obtener las medallas. Pero, ¿qué ocurre si decidimos no retar al líder de un gimnasio y continuar la historia? En sus primeras versiones esto era posible, sin embargo, nada cambiaba si decidías modificar el orden establecido.

A continuación vamos a estudiar qué tipos de técnicas existen para evitar los problemas que hemos descrito.

Capítulo 2

Estado del arte

Hoy en día la Inteligencia artificial todavía está lejos de poder simular el comportamiento del ser humano. No obstante, los avances a lo largo de los últimos años ponen de manifiesto que día a día se mejora la capacidad de los ordenadores para reproducir la mente humana.

A continuación veremos algunas premisas de la inteligencia artificial enfocadas a simular el comportamiento humano y de las técnicas utilizadas en la industria de los videjuegos para resolver problemas similares.

2.1. Premisas de la Inteligencia Artificial

Las primeras ideas relacionadas con la inteligencia artificial surgieron junto a la lógica [4], los algoritmos y las matemáticas [25, 13], los cuales tienen su origen en las culturas griegas o arabes varios años antes de Cristo, pero no es hasta 1950 cuando **Alang Turing** [40] publica un artículo [41], donde se propone una respuesta para considerar a una máquina inteligente, momento en el que se define formalmente el término de inteligencia artificial. Surge así el llamado **Test de Turing** [28].

Según este test, lo que dota a un ser de inteligencia es su capacidad para imitar el comportamiento humano. Para superar el test, la máquina debe ser capaz de engañar al juez para que determine que no es una máquina sino un ser humano.

Esta definición ha ido evolucionando hacia un enfoque donde se pretende imitar el comportamiento racional. Estas ideas son fundamentales dentro del

trabajo que estamos realizando, ya que lo que pretendemos es que el comportamiento de nuestros agentes se aproxime con la mayor fidelidad posible al comportamiento de un ser racional.

Los primeros pasos dados en este campo fueron a través del ajedrez, donde se pretendía que la máquina fuera capaz de ganar a un ser humano. Poco a poco se ha ido evolucionando, llegando a publicarse en 2014 la noticia de que una máquina había conseguido superar el test de Turing [16, 1]. Durante este largo proceso, han surgido conceptos importantes como el aprendizaje, el razonamiento y la percepción del mundo. Si hay uno que podemos destacar, debido a su finalidad de modelar el comportamiento humano en la toma de decisiones es el de agente inteligente.

Un agente inteligente es aquel capaz de percibir el entorno que le rodea y, en base a la información que tiene, tomar decisiones de una manera racional. Los agentes se caracterizan por su capacidad para razonar sobre un determinado tema. Vamos a centrarnos en un tipo concreto de estos agentes: los agentes BDI. Estos agentes, basados en modelos cognitivos, pueden ser los más interesantes para nuestro problema y de gran utilidad en la industria del videojuego.

2.2. Agentes BDI

Existen una gran variedad de teorías y modelos sobre el razonamiento humano, pero de todas ellas la más relevante para nuestro trabajo es la llamada Creencia-Deseo-Intención (BDI del inglés Belief-Desire-Intention).

Este modelo, desarrollado por Michael Bratman [5], está basado en la psicología popular y describe el razonamiento mediante sólo tres características: creencias, deseos e intenciones. El modelo BDI asume que las acciones humanas son el resultado de un proceso llamado razonamiento práctico. Según este proceso, en primer lugar se seleccionan los deseos que se quieren llevar a cabo (objetivos) en función del estado actual en el que nos encontramos. Tras este paso, se establecen qué acciones se van a llevar a cabo para poder alcanzar dicho objetivo.

El modelo de agentes BDI proporciona un mecanismo que permite separar el proceso de elegir el plan que se va a ejecutar de la ejecución de este plan en sí misma, permitiendo calibrar el tiempo empleado entre las decisiones de

un plan y su ejecución.

Según Bratman, tanto intenciones como deseos son características relacionadas con la acción, sin embargo especifica que las intenciones están más relacionadas con la conducta que los deseos, en el sentido del compromiso que el agente tiene con realizar algo. Las intenciones se pueden traducir en un número de pasos que el agente se ha comprometido a realizar para alcanzar cierto objetivo y conseguir satisfacer así un deseo.

Las características que describen este modelo son:

1. **Creencias.** Representan el conocimiento que se tiene del entorno y del mundo. Cambian tras la ejecución de cada acción. También se incluyen las capacidades de deducción de los individuos (reglas de inferencia). Las creencias no tienen porqué ser necesariamente ciertas, ya que se puede tener un conocimiento impreciso o falso del mundo.

2. **Deseos.** Representan el estado emocional de nuestro agente. Guardan la información de los objetivos que se quieren llevar a cabo, así como los costes asociados de conseguir cada uno. Cuando los deseos se activan para intentar lograrlos reciben el nombre de objetivos. Los objetivos, y por lo tanto los deseos, no deben entrar en contradicción entre sí.

3. **Intenciones.** Representan el plan de acción que se ha decidido llevar a cabo. Son la componente deliberativa de nuestro agente. En este momento el agente ha comenzado a ejecutar el plan. Los planes son secuencias de acciones que llevan a la consecución de un objetivo. Los planes se pueden subdividir en planes más pequeños si es necesario llevar a cabo pasos intermedios, por ejemplo para comprar necesito dinero, para tener dinero necesito ir al banco y sacarlo, y para poder sacarlo en el banco necesito la tarjeta de crédito.

4. **Eventos.** Tienen lugar debido a las acciones que realiza un agente. Los eventos pueden modificar las creencias, los objetivos o desatar acciones por parte de los agentes. Los eventos deben ser recopilados por los sensores que el agente tiene sobre el entorno.

Rao y Georgeff adoptaron este modelo para desarrollar agentes software y propusieron un intérprete BDI [35], que es la base de la mayoría de estos sistemas. El intérprete trabaja con las creencias, los objetivos y los planes del agente.

El primero de estos sistemas desarrollado con éxito fue ***Procedural Reasoning Systems* (PRS)** [18] cuyo sucesor fue el llamado ***dMARS*** [10]. El proceso de ejecución de un agente BDI tiene la siguiente estructura:

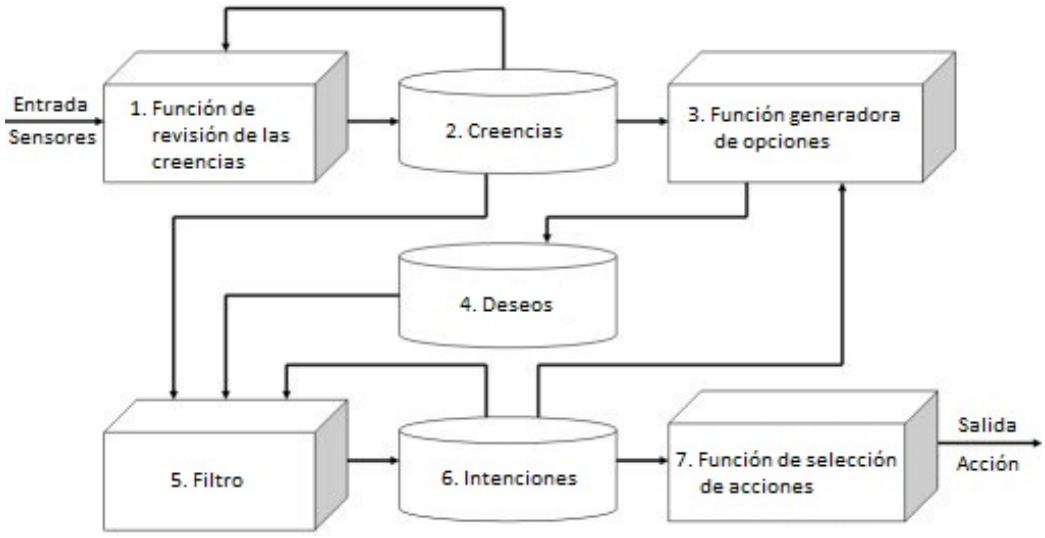

Figura 2.1: Proceso de ejecución de un agente

1. Una función de revisión de las creencias, que tomando como entrada los nuevos datos que se reciben del mundo y las creencias que se tenían anteriormente permiten actualizar convenientemente la imagen que el agente tiene del entorno.

2. Un conjunto de creencias, que representan el conocimiento que el agente tiene del mundo.

3. Una función generadora de opciones, que determina que deseos son posibles para el agente en función del conocimiento que este tiene del entorno.

4. Un conjunto de opciones, los deseos, que representan posibles caminos que el agente va a tomar.

5. Una función filtro que representa el proceso de decisión del agente, según el cual determina sus intenciones en función de las creencias, los deseos y las intenciones que se tenían anteriormente.

6. Un conjunto de intenciones, que representan aquello que el agente quiere alcanzar, es decir, los estados a los que quiere llegar.

7. Una función de selección de acciones, que permite decidir qué acción es la siguiente que vamos a ejecutar basada en la intenciones que tiene el agente.

Podemos implementar un agente BDI desde varios enfoques distintos, sin embargo, en este trabajo, el que más nos interesa es aquel que actualiza su estado de forma continuada y toma decisiones de acuerdo al conocimiento actual. Es por eso que en la siguiente sección veremos cómo implementar agentes BDI que se mueven en entornos dinámicos.

2.3. Agentes BDI en tiempo real: ARTS

Un problema importante a la hora de construir agentes inteligentes, es conseguir que este sea capaz de interactuar en tiempo real con un entorno que varía continuamente. La mayor dificultad reside en cómo especificar las restricciones de tiempo real. En muchas ocasiones, los recursos de tiempo y memoria no son un problema, lo cual ha llevado a desarrollar soluciones sin preocuparse de estas limitaciones. Sin embargo, en entornos de tiempo real, el tiempo que puede llevar tomar una decisión por parte de la IA puede ser tan grande que el entorno ya no es aquel sobre el que empezamos a tomar la decisión.

Arquitecturas como *PRS*[18] y similares resuelven el problema de construir agentes que se desenvuelven en entornos dinámicos. Sin embargo, estas soluciones no son efectivas, ya que carecen de métodos generales para indicar que un cierto objetivo tiene un tiempo límite para ser alcanzado y tienen que ser programados de forma específica. Existen también arquitecturas híbridas (*ROAC* [20] o *SIMBA* [6]), que tienen una IA subyacente y un control de bajo nivel sobre esa IA conectadas mediante una interfaz.

ARTS [43] se propone como un modelo para agentes BDI en tiempo real que tiene en cuenta el tiempo disponible para llevar a cabo una cierta tarea y la prioridad que estas tienen para dicho personaje.

Un sistema de tiempo real puede ser catalogado como crítico (hard real-time) o acrítico (soft real-time). En el primero de ellos el plazo de respuesta debe entrar en un tiempo límite y de no conseguirlo las consecuencias pueden

ser fatales para el resto de la ejecución. El segundo de ellos permite que se puedan producir retrasos a la hora de emitir una respuesta, degradando su utilidad con el tiempo, pero no de forma fatal.

El caso de tiempo real crítico es prácticamente imposible para procesos relacionados con el entorno, solo pudiendo exigirse para cálculos internos del propio personaje. Suponemos que los objetivos tienen un tiempo límite para ser alcanzados, y llamaremos posibles a aquellos que pueden ser alcanzados dentro de dicho tope. No todos los objetivos tienen que ser viables, de hecho los cambios en el entorno pueden hacer que algunos objetivos que eran posibles dejen de serlo o incluso al contrario, aquellos que no lo eran anteriormente puede llegar a serlo.

Es común que algunos objetivos sean más importantes que otros, razón por la que supondremos que estos tienen asociada una prioridad. Estas prioridades definen un preorden ($\preceq$) en los objetivos. Un conjunto de objetivos P se llama maximal si no existe otro conjunto P' distinto tal que $P \preceq P'$. Un agente BDI de tiempo real será aquel que pueda satisfacer más deseos de mayor prioridad.

Un agente BDI consta en general de un conjunto de creencias y deseos. Para cumplir sus objetivos adopta intenciones, que le indican los pasos que debe ejecutar para poder llevar a cabo dichos objetivos. Para resolver casos en tiempo real se debe proporcionar al agente mayor información sobre sus objetivos y limitar el tiempo de ciclo de ejecución del agente.

Es posible que las acciones necesarias para alcanzar ciertos objetivos se subdividan en tareas más sencillas. Estas tareas tienen el mismo tiempo límite que la tarea principal para la que sirven. Sin embargo, es posible que dichas tareas tengan un tiempo límite adicional impuesto por el entorno para poder cumplir la acción. Por ejemplo, suponemos que nuestro personaje tiene que ir al médico. Para ir al médico es necesario pedir con anterioridad una cita, pero las citas solo se pueden obtener en el horario de atención al público, por lo tanto, esto añade una restricción adicional para poder ir al médico.

Cada objetivo tiene una prioridad, por lo tanto los planes llevados a cabo para cumplir este objetivo también tendrán una prioridad asociada. Un agente debe maximizar entre los subplanes que puedan existir aquel que más ayude a la consecución del objetivo.

Cada plan tiene asociada una duración, considerada el tiempo máximo

que puede tardar en ejecutarse. Pasado este tiempo se considera que el plan falló. Un plan que tiene subplanes tiene una duración igual a la suma de todas las componentes necesarias para realizarse.

El cálculo de la duración se puede hacer con el siguiente método:

1. Para cada plan se calculan todas las posibles variantes de una intención dirigidas por ese plan.

2. La información anterior puede ser guardada en forma de árbol, donde las hojas no incluyen llamadas a otros planes. La duración de las hojas se puede calcular en función del tiempo máximo que tiene para ejecutar una cierta acción.

3. A partir de las hojas se puede obtener la duración de todas las variantes de una intención. Todas estas variantes nos proporcionan un límite superior y un límite inferior del tiempo de duración del plan

Si el árbol generado fuera un grafo con bucles, seguimos pudiendo calcular la duración mínima, pero la máxima sería infinita.

En cuanto al tiempo de ciclo del agente, éste debe resolver ciertas tareas como actualizar las creencias y los objetivos. De todas las tareas la selección del plan es particularmente importante, ya que el coste puede ser enorme. Para minimizar lo máximo posible este tiempo debemos ordenar los objetivos y los planes por prioridad. Para nuestro primer objetivo tomamos el primer plan que lo soluciona. Si es factible y entra en tiempo lo llevamos a cabo. En caso de que no se pueda llevar a cabo comprobamos el siguiente en nuestra lista. Seguimos en este proceso hasta que conseguimos establecer un plan para cada objetivo o el tiempo que podemos dedicar a esta tarea ha expirado.

La duración estimada de un plan puede ser calculada de forma optimista (el mínimo tiempo que el agente va a tardar en ejecutarlo) o de forma pesimista (el máximo tiempo que va a tardar). Debido a que pueden existir bucles, duración pesimista será en muchas ocasiones infinita, lo cual proporciona malos resultados. Es por ello que en muchas ocasiones será mejor tomar la duración optimista como referencia.

2.4. Agentes en videojuegos: el ejemplo de F.E.A.R.

Tras hacer un análisis de los agentes BDI y posteriormente centrarnos en aquellos que interactúan con un entorno que varía constantemente (agentes en tiempo real), vamos a ver ejemplos prácticos aplicados en videojuegos. Vamos a centrarnos en el videojuego ***F.E.A.R.*** (*F.E.A.R. First Encounter Assault Recon.* Monolith Productions, 2005), el cual fue una gran revolución, llegando a ser considerado el mejor videojuego de acción del año 2005.

F.E.A.R. es un videojuego de disparos en primera persona y survival horror que conjuga el clásico modo de juego de disparos en primera persona típico de juegos como ***HλLF-LIFE*** (Half-Life, Valve Corporation, 1998) o ***Counter-Strike*** (Counter-Strike, Valve Corporation, 1999) con una atmósfera de terror sobrenatural propio de videojeugos como ***Project Zero*** (Project Zero, Tecmo, 2001), ***Silent Hill*** (Silent Hill, Team Silent, 1999).

La mayoría de videojuegos de este tipo utilizan máquinas de estados finitas (FSM: Finite state machine) para controlar el comportamiento de sus personajes junto al algoritmo A^* para trazar rutas. Sin embargo, este videojuego utiliza estos métodos de un modo distinto.

La máquina de estados de *F.E.A.R.* cuenta con tan solo tres estados:

1. Ir a (GoTo)
2. Ejecutar animación (Animate)
3. Usar un objeto inteligente (UseSmartObject)

Un objeto inteligente es un tipo concreto de objeto que almacena el conocimiento necesario para que el agente pueda ejecutar su tarea. Esto quiere decir que, a efectos prácticos, un objeto inteligente es similar a una animación propiciada por el objeto, razón por la que solo vamos a considerar los estados GoTo y Animate.

En efecto podemos reducir todo el comportamiento de un agente a moverse y ejecutar animaciones. Por ejemplo atacar a un enemigo consiste en ejecutar repetidamente la animación de disparar. La decisión importante reside por tanto en encontrar cuando debe realizarse una transición entre los dos estados posibles. Esta decisión queda definida por el programador, que

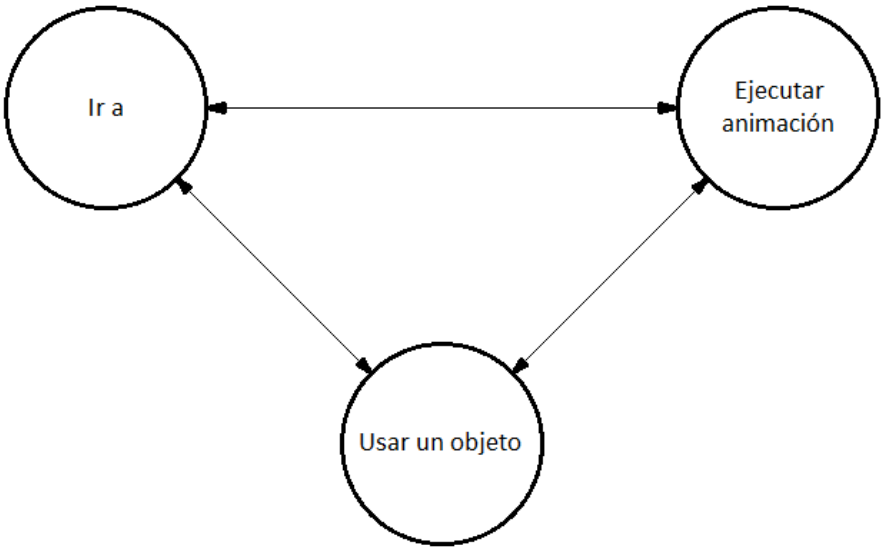

Figura 2.2: Máquina de estados de *F.E.A.R.*

especifica estos cambios mediante un sistema de planificación.

Los videojuegos anteriores de este género eran bastante más básicos. En un principio, la IA de estos videojuegos se encargaba únicamente de avisar de avistamientos y disparar (***Shogo***, Monolith Productions, 1998). Posteriormente se añadió conocimiento para cubrirse del fuego enemigo y saber cuando disparar (***No One Lives Forever***, Monolith Productions, 2000). *F.E.A.R.* añade nuevas funcionalidades teniendo en cuenta la colaboración entre varios personajes (coberturas en tiroteos).

La complejidad para abordar estos avances es demasiado grande, por esta razón se debe cambiar el paradigma que se tenía hasta entonces para desarrollar la IA. Las máquinas de estados marcan el comportamiento exacto que deben tener los personajes mientras que el sistema de planificación proporciona una lista de objetivos que deben satisfacer, dejando a la IA el proceso

de como resolverlo.

El sistema de planificación consiste en la búsqueda de una secuencia de acciones para alcanzar estos los objetivos. Este sistema de planificación es muy parecido a **STRIPS** [14].

STRIPS (**St**anford **R**esearch **I**nstitute **P**roblem **S**olver) es un generador de planes automatizado. STRIPS se basa en objetivos y acciones, donde los objetivos son estados que queremos alcanzar y las acciones se caracterizan por unas precondiciones necesarias y los efectos que se producen. Una acción solo se puede producir si todas sus precondiciones se cumplen y cambian el estado de un cierto modo.

Cada acción tiene asociada una serie de cambios, que se traducen en eliminar cierto conocimiento del mundo y añadir otro en su lugar. No se puede simplemente modificar el conocimiento que se tiene, ya que no se puede garantizar ni la existencia ni la unicidad de un cierto conocimiento. Por ejemplo, puede estar lloviendo o no llover, pero no pueden ocurrir ambos fenómenos a la vez. De igual manera, es posible que un mapa tenga árboles, pero también puede tener casas y vehículos.

La principal ventaja del sistema de planificación es separar el trabajo de diseñador gráfico, encargado de crear los entornos óptimos para la historia del videojuego y de las acciones válidas por cada personaje, de la tarea de describir el comportamiento concreto de cada personaje en cada situación.

De esta manera el diseñador solo tiene que establecer las acciones que pueden ejecutar cada uno de los distintos personajes del videojuego, mientras que el comportamiento que verdaderamente van a tener queda descrito por sus objetivos.

Diferentes personajes podrían tener los mismos objetivos, pero cada uno los resolvería de un modo diferente en función de las posibles acciones que puede ejecutar y de su comportamiento. Por esta razón, si tenemos un fusilero y un francotirador que desean acabar con un objetivo, llevarán a cabo diferentes acciones para terminar su misión.

Anteriormente, para llevar a cabo estos propósitos, era necesario crear complejas máquinas de estados en los que se tenían en cuenta los diferentes casos, ya que no era posible separar las acciones del objetivo que se quería alcanzar. Un ejemplo de esto se da en el videojuego ***No one lives forever***

2 (No One Lives Forever 2: A Spy in H.A.R.M.'s Way, Monolith Productions, 2002) donde los policías que no están en forma deben pararse para recuperar el aliento. Aunque este caso se da en muy pocas ocasiones (de hecho basta solo con una) se debe modificar las máquina de estados para tenerlo en cuenta.

Separar ambos conceptos permite además añadir nuevos personajes de forma muy rápida, sencilla y sin poner en riesgo el correcto funcionamiento de los otros personajes que ya se tenían anteriormente.

La segunda ventaja que se obtiene de este sistema, es poder separar el comportamiento en varias capas, para así poder reutilizarlo y tener personajes similares pero con pequeñas variaciones.

En el videojuego *F.E.A.R.* un soldado tiene hasta 7 capas en su comportamiento:

- El más básico es la capacidad de atacar a un enemigo cuando lo detecta, con lo que satisface el objetivo de Eliminar al enemigo.
- El segundo, consiste en valorar su vida, para lo cual intenta esquivar los disparos que le lancen.
- El tercero es reaccionar con combate cuerpo a cuerpo cuando el enemigo está demasiado cerca, otra acción que satisface el objetivo de Eliminar al enemigo.
- El cuarto es cubrirse para realmente asegurarse de que valora su vida. Dentro de esta capa se añaden las características de atacar desde una posición cubierta y esquivar los disparos desde una posición cubierta.
- El quinto consiste en poder atacar sin salir de nuestra zona de cobertura, de modo que añadimos el fuego a ciegas.
- El sexto es detectar cuando nuestro escondite deja de ser seguro y es necesario refugiarse en otro sitio distinto.
- El séptimo es la capacidad de "diálogo" para comunicarse con sus compañeros de equipo que además permite al jugador saber que piensa la IA

La mejora que supone este método es que no tenemos que especificar concretamente qué acción se debe realizar para alcanzar nuestros objetivos. Los

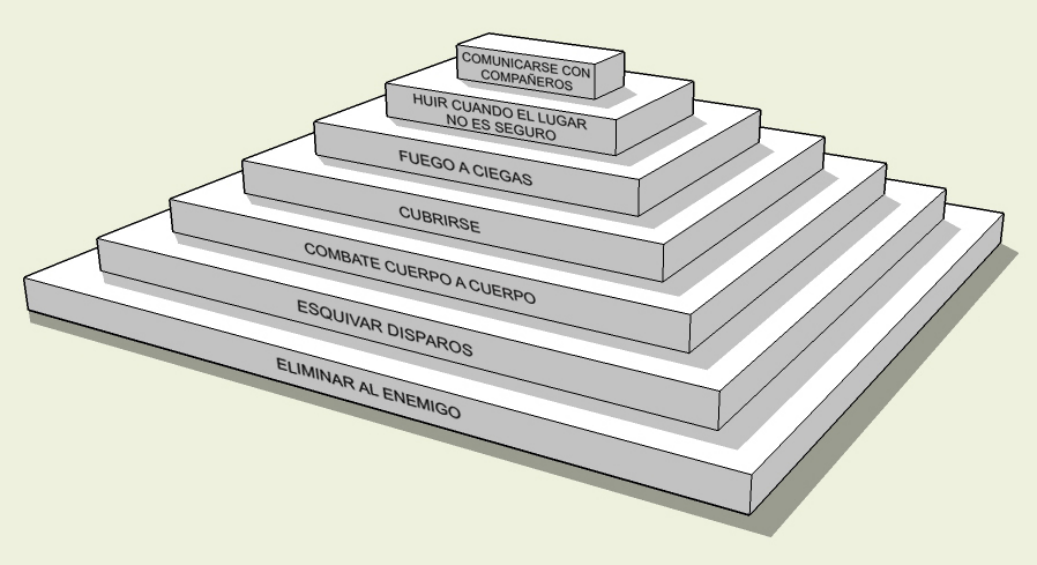

Figura 2.3: Capas de comportamiento de un soldado en el videojuego F.E.A.R.

objetivos tienen una serie de precondiciones necesarias para poder realizarse y las acciones tienen un conjunto de efectos derivados de su ejecución. La IA unifica las acciones necesarias en función de sus efectos para poder cumplir las precondiciones y así satisfacer los objetivos propuestos.

Una última ventaja que supone el sistema de planificación es la resolución dinámica de los problemas. Esto nos permite poder replanificar las acciones que vamos a llevar a cabo para alcanzar nuestro objetivo. Si una cierta acción no puede ser llevada a cabo, por ejemplo porque el camino que queremos seguir está obstruido, buscaremos una solución alternativa ahora que tenemos un mayor conocimiento del mundo (otras ruta hasta nuestro objetivo).

Como se indicó anteriormente, el sistema de planificación de F.E.A.R. difiere ligeramente de STRIPS. Este sistema, al que nos referiremos como **GOAP** (del inglés Goal-Oriented Action Planning) [30], hace el método mucho más útil para la toma de decisiones en tiempo real. Se distinguen 4 diferencias principales entre GOAP y STRIPS:

1. **El coste por acción.** Para conseguir que los personajes prefieran llevar a cabo ciertas frente a otras se añade un coste a cada una de ellas.

Por tanto para satisfacer un objetivo nuestro personaje debe buscar las secuencia de acciones con menor coste que permite alcanzarlo.

Para elegir la mejor acción se puede utilizar el algoritmo A^*. En este caso los diferentes estados del mundo son los nodos del grafo y los costes son los de la acción que nos mueve entre los diferentes estados. El algoritmo debe encontrar el camino óptimo entre el estado inicial y el estado objetivo al que queremos llegar. El algoritmo A^* también es utilizado para encontrar el camino óptimo entre dos lugares.

2. **La eliminación de las Listas de Añadir y Borrar para efectos.** En vez de tener una lista de conocimiento a añadir o eliminar como efecto de las acciones vamos a utilizar un array de tamaño variable que representa el estado del mundo.

3. **Añadir procedimientos a las precondiciones.** Dado que no podemos representar completamente el juego, podemos añadir controles para especificar cuándo debemos ejecutar un procedimiento. Por ejemplo, si en el desarrollo del juego se va a producir una explosión, es mejor salir corriendo que buscar un refugio, sin embargo el coste de buscar este refugio y llegar a él puede ser demasiado grande, por lo que se debe evitar en la medida de lo posible.

4. **Añadir procedimientos a los efectos.** Dado que los cambios de estados no son instantáneos sino progresivos (por ejemplo acabar con un enemigo), tratamos los diferentes cambios que se producen como una máquina de estados. Según vamos ejecutando acciones para alcanzar nuestro objetivo, el mundo cambia de forma progresiva avanzando por lo diferentes estados.

2.5. Lógica subjetiva

Hasta ahora nos hemos centrado en cómo desarrollar personajes que perciben el entorno en el que se mueven y deciden qué acciones deben realizar, pero no hemos hablado de qué podemos hacer para manejar el conocimiento que tenemos. Nuestro personajes deben razonar para llegar a conclusiones a partir de lo que otros personajes han dicho, pero este conocimiento al que llegamos no es del todo fiable, razón por la que vamos a tener que tratar con un conocimiento del que tenemos desconfianza o incertidumbre.

La lógica subjetiva es un tipo de lógica probabilística que tiene en cuenta la incertidumbre y creencias. Es adecuada para modelar situaciones de incertidumbre o de conocimiento incompleto [22, 23]. Puede ser usada por ejemplo para modelar redes de verdad o hacer análisis de redes bayesianas.

Los argumentos son "opiniones subjetivas" sobre proposiciones. Una opinión binomial se aplica sobre proposiciones simple y puede ser representado mediante una distribución beta [Anexo B: Distribución Beta]. Las opiniones multinomiales son aplicadas sobre una colección de proposiciones y puede ser representado mediante una distribución de Dirichlet [Anexo C: Distribución Dirichlet].

Dado que el ser humano no puede afirmar algo sobre el mundo real con absoluta certeza, la validez de una proposición no puede ser considerada objetiva. De hecho, cualquier afirmación es expresada por algún individuo concreto cuya percepción sobre la realidad puede ser diferente de la de cualquier otro (por ejemplo un daltónico no puede distinguir los colores como alguien que no lo sea, pudiendo considerar que el otro está equivocado), de modo que no se puede tomar como conocimiento objetivo.

Las opiniones subjetivas expresan creencias subjetivas sobre la verdad de las proposiciones con grados de incertidumbre o creencias propias del sujeto. Las opiniones se denotan como w_x^A (también escrito a veces con la notación $w\,(A : x)$), donde:

- $\boldsymbol{A}$ es el sujeto, también llamado propietario del conocimiento (belief owner).

- $\boldsymbol{x}$ es la proposición sobre la que se opina

La proposición x pertenece a un marco de criterio o espacio de estados (usualmente se denota por X) pero no se suele incluir en la notación. Las proposiciones de un marco suelen tomarse totalmente disjuntas y se supone que todos los sujetos interpretan las proposiciones de forma similar, es decir, tienen la misma percepción del mundo.

Dada una proposición x, una opinión binomial es una tupla $w_x = (b, d, u, a)$ donde:

b	creencia (belief)	creencia de que la proposición es cierta
d	desconfianza (disbelief)	creencia de que la proposición es falsa
u	incertidumbre (uncertainty)	es el conocimiento no contrastado, lo que no se sabe con seguridad
a	tasa básica (base rate)	es la probabilidad a priori de veracidad de una afirmación en ausencia de evidencias

Estas componentes satisfacen que: $b + d + u = 1$ donde $b, d, u, a \in [0, 1]$

Una opinión binomial puede representar con facilidad elementos de la lógica tradicional o de la lógica probabilística. Por ejemplo:

$b = 1$	es equivalente a cierto en lógica binaria
$d = 1$	es equivalente a falso
$b + d = 1$	es equivalente a la lógica probabilística tradicional

La esperanza (en sentido matemático) de que una opinión sea cierta se define como:

$$E = b + au$$

Las opiniones binomiales se pueden representar mediante un triángulo, donde cada punto interior del triángulo es una tupla (b, d, u). Los ejes *b, d, u* van de un vértice al contrario etiquetados con los valores de Creencia, Desconfianza e Incertidumbre.

La tasa básica (también llamada atomicidad), es un punto de la base del triángulo (un punto entre los vértices de Creencia y Desconfianza). La esperanza de una opinión, E se obtiene proyectando sobre la base una paralela a la recta que pasa los puntos de tasa básica y el vértice de Incertidumbre.

La distribución Beta de una opinión binomial $w = (b, d, u, a)$ es la función $Beta\,(\alpha, \beta)$ donde:

$$\alpha = 2b/u + 2a$$
$$\beta = 2d/u + 2\,(1 - a)$$

En la representación anterior, la proposición X es muy probable y tiene un pequeño grado de incertidumbre. La proposición Y tiene aproximadamente una probabilidad del 50 % y es incierta. Por último la proposición Z es

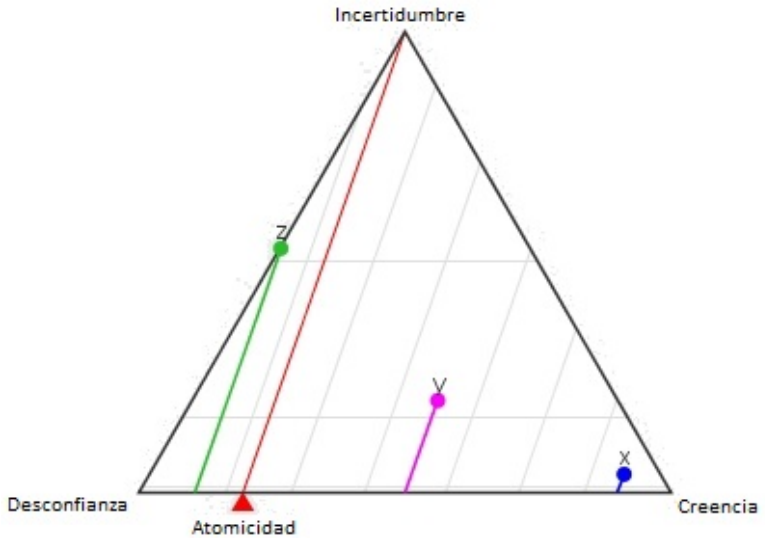

Figura 2.4: Representación de 3 proposiciones

muy improbable y muy incierta.

Dado un marco X de proposiciones disjuntas x_i, una opinión multinomial sobre X es la función $w_x = \left(\vec{b}, u, \vec{a}\right)$ donde $\vec{b}$ es el vector de creencia sobre las proposiciones de X, u es la incertidumbre y $\vec{a}$ es el vector de tasas básicas sobre las proposiciones de X.

Las componentes satisfacen:

$$u + \sum \vec{b}\left(x_i\right) = 1 \qquad\qquad \sum \vec{a}\left(x_i\right) = 1$$

$$\text{donde } \vec{b}\left(x_i\right), u, \vec{a}\left(x_i\right) \in [0, 1]$$

Las opiniones multinomiales no se pueden representar de una forma sencilla como las binomiales. La representación consiste en elementos similares al triángulo pero de dimensiones superiores. De este modo, para opiniones trinomiales tendríamos una pirámide de base triangular. Para representar opiniones de mayor dimensión debemos aumentar la dimensión del espacio sobre el que la representamos convenientemente.

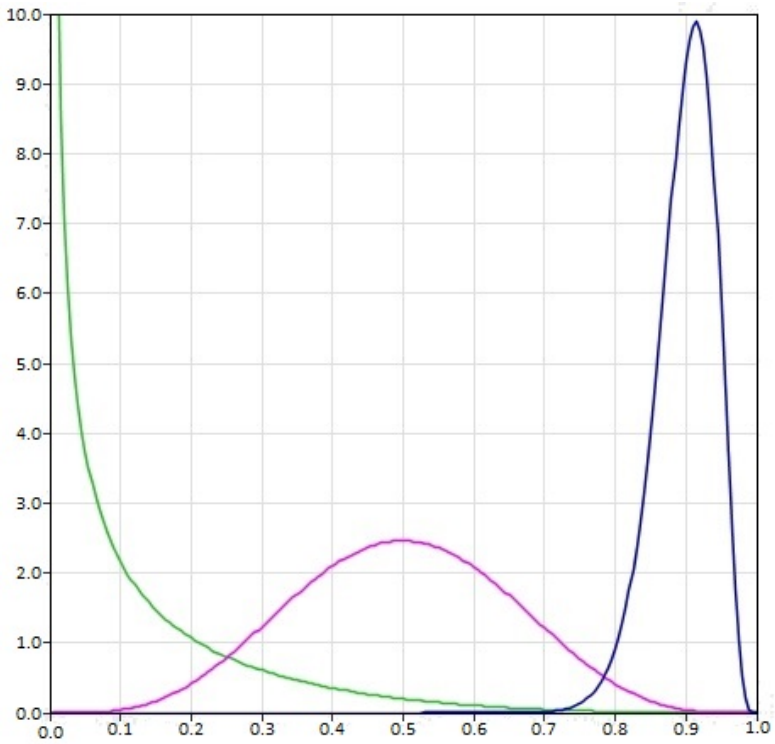

Figura 2.5: Distribución Beta asociada a las 3 proposiciones

La distribución de Dirichlet de un opinión multinomial $w_x = \vec{b}, u, \vec{a}$ es la función $Dir\left(\vec{\alpha}\right)$ donde $\vec{\alpha}\left(x_i\right) = 2\vec{b}\left(x_i\right)/u + 2\vec{a}\left(x_i\right)$

La lógica subjetiva es útil para analizar casos donde la incertidumbre y el conocimiento incompleto juegan un papel importante. Esto permite ver la lógica subjetiva como una lógica probabilística aplicada a casos con probabilidades inciertas. Como dijimos anteriormente la lógica subjetiva es de gran utilidad en redes de verdad y redes bayesianas.

Veamos un ejemplo de su aplicación en una red de verdad [24]: Suponemos que tenemos cuatro personas (A, B, C, D), de modo que A tiene una cierta confianza en B y una cierta confianza en C. Estos dos últimos tienen cada uno su propia confianza en D. Entonces, tanto B como C pueden proporcionar a A información sobre D de modo que A tendrá una confianza en D dada por:

$$w_D^A = \left(w_B^A \times w_D^B\right) \oplus \left(w_C^A \otimes w_D^C\right)$$

Las redes de verdad nos permiten indicar la fiabilidad del origen de las proposiciones, permitiéndonos obtener opiniones subjetivas sobre dichas proposiciones.

2.5.1. Lógica subjetiva basada en evidencias

La lógica subjetiva basada en evidencias (EBSL del inglés Evidence-base subjetive logic) es una variante de la lógica subjetiva en la que la transitividad de las opiniones se mantiene aplicando pesos a las evidencias que soportan las opiniones. La regla de transitividad en esta lógica permite realizar redes de verdad arbitrarias.

Dada una proposición P, si llamamos p al peso de la evidencia que respalda a P y n al peso de la evidencia que respalda que P sea falso ($\neg P$), podemos representar la evidencia como un vector (p, n). Sea c una constante positiva que representa la unidad de evidencia. Dada una opinión (b, d, u) cuya base es la evidencia (p, n), entonces existe una función $1-1$ entre dicha opinión y la evidencia dada por:

$$(b, d, u) = \frac{(p, n, c)}{p + n + c} \qquad (p, n) = c\frac{(b, d)}{u}$$

Esta función es la solución de las siguientes restricciones:

$$b/d = p/n$$
$$b + d + u = 1$$
$$p + n = 0 \Longrightarrow u = 1$$
$$p \to \infty \Longrightarrow u \to 0$$

Igual que en la lógica subjetiva, podemos representar los valores de una proposición dentro de un triángulo. Las opiniones que caen sobre el segmento entre los puntos de Creencia y Desconfianza (Incertidumbre = 0), son llamadas opiniones dogmáticas. Esto solo es alcanzable con una cantidad infinita de evidencias, razón por la que a menudo suelen excluirse.

Una vez explicadas estas ideas relativas a los agentes inteligentes y la manera en que pueden gestionar el conocimiento, pasamos a exponer los objetivos del trabajo y a presentar cómo hemos aplicado lo aprendido para plantear un problema interesante en el ámbito de los videojuegos y un modelo computacional que trata de establecer un marco de resolución para el mismo.

Capítulo 3

Objetivos

El objetivo de este trabajo es proponer una solución a los dos problemas planteados en la Introducción: evitar tener personajes predeterminados que no razonan sobre su comportamiento y utilizar el conocimiento adquirido para que puedan responder de forma razonada a preguntas.

No tratamos de resolver un problema concreto, sino buscar una solución a un conjunto de problemas generales que pueden estar presentes en una variedad de videojuegos prácticamente infinita y cuya complejidad no está acotada. Es por esto que debemos definir un entorno lo más reducido posible pero que mantenga intactos los elementos que caracterizan al reto al que queremos hacer frente.

Dado que debemos tratar con personajes que interactúan unos con otros compartiendo información, de la cuál no se tiene certeza de su veracidad o falsedad, vamos a definir el juego como un videojuego multijugador de detectives. Este juego presenta una familia de posibles escenarios donde queremos seguir trabajando en el futuro. Para probar las ideas aportadas se plantea el siguiente escenario concreto.

3.1. El laboratorio nipón y los roedores

Nos encontramos en un laboratorio de experimentación animal de una universidad japonesa donde trabaja un anciano investigador en psicología. En dicho laboratorio hay un laberinto en el que habita un grupo de traviesos ratones bajo la estricta vigilancia del investigador, que les trata como a sus empleados. Durante el día y mientras no reciban órdenes contrarias, los ra-

tones permanecen inmóviles en sus correspondientes esquinas del laberinto; pero por la noche, cuando el científico duerme, aprovechan para corretear libremente por el laberinto y satisfacer el deseo de jugar entre ellos o incluso comer del queso adulterado que se encuentra en el centro de la instalación, algo terminantemente prohibido en este laboratorio, ya que arruinaría los experimentos en curso.

Este alboroto puede realizarse, pero no sin tomar ciertas precauciones ya que el científico a la mañana siguiente vendrá a valorar la situación. El investigador confía en que los ratones se encuentren en su ubicación original, que las paredes del laberinto más frágiles por estar fabricadas en papel (*shoji*) permanezcan intactas, y por supuesto que el apetecible manjar central esté intacto. De no ser así, el anciano comenzará un interrogatorio entre los ratones (usando una técnica avanzada de comunicación animal inventada precisamente en este laboratorio) para averiguar la verdad sobre lo ocurrido y castigar con penas proporcionales al culpable o los culpables de las fechorías perpetradas.

Además el científico realizará pruebas a los ratones para comprobar su evolución. Debido a que les ha inoculado diversos fármacos experimentales, si los ratones han jugado entre sí o con los juguetes esparcidos por el laberinto, estas pruebas proporcionarán unos resultados alterados, lo que le llevará a buscar que ratones son los responsables.

El juego por lo tanto consta de dos fases fuertemente diferenciadas, una que tiene lugar durante la noche, cuando el científico duerme en la que los ratones realizan sus travesuras y otra, que ocurre por la mañana tras esta primera fase, en la que se transcurre el interrogatorio hacia los ratones por parte del científico.

3.2. Movimiento y acciones de los ratones

En primer lugar se quiere evitar que nuestros personajes actúen de forma predeterminada y en su lugar razonen que acciones son las mejores en función de los hechos que ocurren durante la ejecución. Es por ello que tenemos que desarrollar una inteligencia capaz de controlar el movimiento y actuaciones de los ratones por el tablero.

Las acciones serán guiadas por una serie de deseos, que varían de unos

ratones a otros. Los ratones también pueden tener sentimientos hacia otros ratones, que no necesariamente tienen que ser correspondidos. Estos deseos y sentimientos serán traducidos en intenciones, dando lugar a las siguientes posibilidades:

1. Buscar y comerse el queso
2. Buscar a otro ratón para jugar con él (puede que el otro ratón no quiera jugar con nosotros).
3. Caminar por el tablero
4. Permanecer quieto

Los ratones siempre van a intentar satisfacer todos los deseos posibles, empezando por aquellos que son más valiosos para ellos. Sin embargo, la prioridad de un deseo puede cambiar dinámicamente si los eventos que tienen lugar durante la ejecución facilitan poder alcanzar otro deseo o ponen en riesgo al ratón, al exponerlo frente a posibles acusaciones en el interrogatorio.

Hemos mencionado que existen elementos frágiles que se pueden romper debido a los actos de los ratones en el tablero, llamados shojis. Los shojis son un tipo de puerta tradicional japonesa, que consiste en una lámina de papel japonés (fabricado con plantas de flora local) cubierto de un marco de madera. A su paso por los shojis, los ratones rompen estas puertas, dejando evidencias de que alguien ha estado en ese lugar. Los ratones deben jugar con la posibilidad de romper o no dichos elementos con la finalidad de alcanzar nuestros objetivos o inculpar a otros ratones.

Con este fin se va a desarrollar una IA que recibe como entrada una serie de deseos que tiene que cumplir, o que desea cumplir lo máximo posible. Cada uno de estos deseos tiene un peso específico, que indica cómo de valioso es ese deseo para el ratón.

Además de sus deseos el ratón conocerá su color, la posición que ocupa en el tablero y tendrá información de todo lo que ha visto a lo largo de la ejecución. La información de lo que ha visto anteriormente es muy útil, porque nos ayudará a encontrar nuestros posibles objetivos. Con esta información decidiremos cual es la siguiente acción que mejor optimiza nuestros objetivos.

Para satisfacer los deseos de manera óptima se hará una búsqueda por todos los datos conocidos del tablero. Se calculará la distancia aproximada

mediante una heurística y se ordenarán los objetivos con una nota que es una ponderación entre la distancia del ratón a su objetivo y el peso que tiene el deseo que le lleva a querer alcanzarlo.

De esta lista se escogerá el primero y se buscará la forma óptima para conseguirlo, usando el algoritmo A^* en la búsqueda de nuestro objetivo. Existen deseos que se pueden cumplir de forma simultánea, algo que se va a manifestar en distintas implementaciones del algoritmo A^*. Por ejemplo un ratón puede desear andar y comerse el queso. Mientras camina hacia el queso estaría cumpliendo simultáneamente ambos objetivos. Algo similar ocurre en los casos que vamos a ver a continuación.

Hemos mencionado que el ratón puede no querer romper los shojis. Esto se reflejará como un deseo del propio ratón de no romper nada, y modificará como se mueve por el tablero, pero nunca lo va a desviar de sus objetivos principales, que serán comerse el queso o jugar con otro ratón. En un principio, el algoritmo A^* trata los shojis como casillas transitables, sin embargo, existe una versión alternativa, que los trata como casillas bloqueadas a las que no se puede acceder. Esta versión se utiliza cuando queremos evitar pasar por los shojis, pues la penalización por romperlos es muy alta.

Otro caso en el que podemos cumplir varios objetivos simultáneamente se da si el ratón tiene el deseo de acabar la ejecución en el punto desde el que salió, aunque en ese caso deberá tener en cuenta el tiempo necesario para volver a casa.

Cuando un ratón llega a la casilla deseada, ejecuta la acción correspondiente.

3.3. Interrogatorio: preguntas y respuestas

En una segunda fase, también deberá ser capaz de entablar un "diálogo", en un lenguaje muy restringido, con el científico para contestar a las preguntas que este le realice. Dentro del diálogo el ratón debe ser capaz de entender lo que pasa por la mente del científico para poder evitar que se le acuse de haber cometido irregularidades y sufrir por tanto castigos.

A partir del conocimiento obtenido durante la ejecución, los personajes serán capaces de razonar y argumentar una defensa de lo ocurrido con el fin

de sacar a la luz la verdad en caso de que sean inocentes o engañar al detective que intenta castigarlos si son culpables. Para conseguirlo, razonarán quien de todos los ratones tiene más posibilidades de salir culpable e intentarán hacer ver al científico que efectivamente ese ratón ha cometido el crimen.

Los ratones recrean el razonamiento del científico en su mente de manera que suponen lo que está pensando y cómo les va a acusar para poder elegir la mejor respuesta. Las pruebas aportadas por lo ratones pueden ser imprecisas o incluso inventadas.

Para poder mantener una comunicación entre varias agentes es necesario especificar un lenguaje que todos entiendan. Este lenguaje debe ser lo suficientemente amplio para que todas las posibles preguntas que el científico necesite hacer estén contempladas.

En esta misma parte está la mente del científico, el cual debe tener un modo de iniciar el interrogatorio para intentar averiguar quién o quienes han cometido los crímenes. A medida que avanza el interrogatorio el científico debe reforzar o debilitar la información recibida en función de la confianza que tiene sobre cada ratón, para finalmente acusar a aquel cuya defensa no parece razonable.

El científico preguntará a los ratones por ellos mismos o por otro ratones, para así poder tener cierta información de las acciones que tuvieron lugar durante la noche. El científico tendrá un nivel de confianza en cada ratón que a priori no tiene que ser el mismo para todos. La confianza de un científico en cada ratón podrá variar a medida que avanza el interrogatorio.

Capítulo 4

Metodología y plan de trabajo

Dado que en la industria del videojuego no existen juegos con una inteligencia artificial que supongan un verdadero reto para el jugador se decide investigar las causas y sus posibles soluciones. En concreto nos centramos en videojuegos de misterio con personajes que realmente parezcan inteligentes. El problema que resultaba más interesante era aquel que trataba personajes con la capacidad de mentir que pudieran construir sus propios argumentos mediante el razonamiento y cómo tratar estos juicios de dudosa veracidad.

4.1. Plan de trabajo

Una vez detectado el problema que se quería resolver, debíamos conocer los métodos más llamativos que se utilizan en las historias sobre detectives. Para abordar dicho problema desde un punto de vista lo más realista posible comenzamos viendo series de detectives, donde debían resolver crímenes.

4.1.1. Obtener información de series

Comenzamos con esta tarea en ***octubre de 2015***. De la gran variedad de series que podemos encontrar, nos centramos en la serie Colombo [26] por sus complejas tramas, aunque también tomamos ejemplos concretos de series más sencillas como Sherlock Yack [3].

La serie Colombo se centra en el personaje Lieutenant Columbo, un policía del departamento de homicidios de Los Ángeles. En la serie, el detective Columbo debe descubrir al responsable del crimen. En el transcurso de un

capítulo destacamos por una parte que todos los criminales dejan pistas sobre su identidad, razón por la que es necesario inspeccionar el lugar, y que interrogando al sospechoso de manera inocente podemos llegar a engañarlo para que se termine inculpando.

Por otra parte, Sherlock Yack es una serie infantil con casos muy simples lo que hace que sean más sencillos de modelar y por lo tanto más interesantes en nuestro trabajo. En este caso tenemos un detective (que además es el gerente del zoo) y su aprendiz que deben resolver el crimen. A lo largo del episodio se detallan pistas y pruebas sobre quien es el posible culpable. El número de personajes que aparecen en cada episodio suele ser reducido.

A partir de estas series se trató de sacar una serie de similitudes con los juegos actuales y se razonó qué elementos faltan en los videojuegos que no los hacían tan creíbles como esperamos. La conclusión fue que estos personajes son demasiado planos en los videojuegos: siguen un guión preestablecido pero no razonan sobre lo ocurrido y cambian sus acciones y los argumentos para defenderse. Esto nos llevó a intentar desarrollar una solución.

4.1.2. Estudiar problemas relacionados

Con información suficiente sobre cómo estructurar el posible videojuego, pasamos a estudiar problemas relacionados con el nuestro. Esta tarea se llevó a cabo de ***noviembre de 2015*** a ***enero de 2016***. Con ayuda de esta información se empezó a escribir el capítulo del Estado del arte. Conociendo las distintas soluciones que podían servir de ayuda en nuestro trabajo y con información sobre como plantear una solución interesante pasamos a definir el juego sobre el que se trabaría y se realizarían las pruebas.

4.1.3. Definir el juego

Con la finalidad de simplificar el problema se definió el juego *El laboratorio nipón y los roedores*, el mes de ***enero de 2016***. Lo que se pretende es crear personajes simples de los que no se espere una gran capacidad de actuación, pero que todo aquello que hagan sea de forma razonada y con un objetivo. Esto permitiría añadir a los ratones la capacidad de mentir y argumentar de manera atómica, sin separarnos de la idea de obtener un videojuego agradable y entretenido. El objetivo de los ratones durante el interrogatorio es salvarse del castigo. Esto los lleva a actuar de forma intencionada para poder

satisfacer sus deseos sin ser pillados. En el interrogatorio los ratones mienten por una importante razón y es que no quieren ser declarados culpables.

4.1.4. Detallar los objetivos

Definido el juego sobre el que íbamos a desarrollar la solución, se realizó el capítulo de la memoria correspondiente a los objetivos. Se formalizaron los problemas que debe resolver la solución propuesta y las distintas partes que va a tener la solución a cada uno de los problemas.

Tras tener una idea de lo que se quería realizar se presentó de manera informal una primera versión de la solución. En esta primera versión los ratones podían moverse libremente por el tablero y podían portarse mal de varias maneras: comiéndose el queso, visitándose entre ellos, tirando probetas del laboratorio o activando las trampas que los deja atrapados.

4.1.5. Describir un modelo formal del juego

Posteriormente este modelo se detalló formalmente y se simplificó de cara a poder hacer una primera versión más rápidamente, la cual posteriormente se iría mejorando. La primera vesión del modelo se obtuvo en ***febrero de 2016***. El modelo establece las características del tablero y los ratones, y el funcionamiento tanto de la fase de simulación como de la de preguntas. En el primero modelo disponemos de un tablero de $N \times M$ con 4 ratones que pueden verlo todo. El flujo de ejecución es el primer ratón actúa y todos los ratones observan, luego el segundo ratón actúa y todos observan. Se continúa así hasta que todos los ratones han actuado una vez y se vuelve al inicio del bucle.

Inicialmente se esperaba que la inteligencia artificial se conectara con una interfaz gráfica a través de un estándar descrito en el modelo correspondiente a la versión. Esto finalmente no se llegó a implementar, pero estableció las bases de la metodología que se iba a seguir en el desarrollo.

4.1.6. Desarrollar los agentes inteligentes

Fueron necesarias varias etapas antes de conseguir un desarrollo definitivo de los agentes inteligentes. Los distintos ciclos por los que pasó fueron:

1. Escribir el código relativo a agentes que tienen como deseos comerse un queso, jugar con otros ratones o moverse. Este código, que se completó en ***marzo de 2016***, consiste principalmente en un algoritmo que toma las ideas de agentes inteligentes que hemos visto previamente.

2. Probar el comportamiento de los agentes inteligentes. Era necesario comprobar que los agentes seguían una conducta lógica propia de un ser inteligente de acuerdo a los deseos que tenían. Las pruebas se hicieron en el mes de ***abril de 2016***. Los resultados mostraron que la implementación no era correcta, dando lugar a situaciones absurdas como que el ratón que se comía el queso se quedara en esa casilla el resto de la partida.

3. Describir un nuevo modelo. En ***mayo de 2016*** se especificó un nuevo modelo teniendo en cuenta los errores detectados anteriormente. Fue necesario añadir nuevos deseos que evitaran comportamientos poco acordes a un ser racional que efectivamente se ha comportado mal.

4. Modificar la impliementación de los agentes para mejorar su comportamiento. En ***julio de 2016***, ya definidos los cambios que se querían hacer, se realizó una nueva implementación de los agentes teniendo en cuenta los problemas de las pruebas. Con estos cambios se llevaron a cabo nuevas ejecuciones deprueba y los resultados fueron satisfactorios.

4.1.7. Desarrollar la mente del científico y de los ratones

En el desarrollo de la mente tanto del científico como de los ratones fueron necesarias de nuevo varias etapas. Estas son:

1. Escribir el código correspondiente a la mente del científico en ***junio de 2016***. Se estableció un plan que el científico debía llevar a cabo para poder obtener pistas sobre el responsable de haber arruinado sus experimentos.

2. Escribir el código concerniente a la mente de los ratones en ***julio de 2016***. Se hicieron varias mentes de prueba y finalmente se desarrolló una con la capacidad de mentir para no ser acusados de haber cometidos el crimen.

3. Pruebas de las distintas mentes en un interrogatorio. En ***agosto de 2016*** se realizaron pruebas de las distintas mentes, haciendo que los ratones contestaran a las preguntas formuladas por el científico. Los resultados mostraron que los ratones no sabían mentir, razón por la que el científico acertaba la mayoría de las veces. Fue necesario realizar una segunda versión de la mente de los ratones, llevandonos a añadir mejoras a la mente del científico para evitar que los ratones le engañaran siempre.

4.1.8. Redactar y revisar los distintos capítulos de la memoria

Finalmente en ***agosto de 2016*** se redactaron los capítulos restantes de la memoria. En esta fecha se realizaron también las convenientes modificaciones y correcciones a la memoria.

4.2. Metodología utilizada

Dado que la complejidad que puede llegar a alcanzar un sistema de conversación es exponencialmente grande, se decidió partir de un primer modelo muy básico, con ratones que tienen muy pocos deseos y que no son capaces de decir mentiras muy complejas, e ir incrementándolo poco a poco.

Se tomó por lo tanto un desarrollo ágil de software. El desarrollo ágil de software describe una serie de principios de desarrollo de software basados en el desarrollo iterativo e incremental gracias al esfuerzo de equipos autoorganizados y multidisciplinares. La elección de este tipo de desarrollo se tomó por una clara razón: se querían obtener resultados simples de una posible solución a los problemas planteados. Los resultados necesitaban poderse obtener rápidamente para posteriormente ir complicando la historia y hacer más sofisticadas las mentes tanto de los ratones como del científico.

Tomando como referencia lo estudiado en el Estado del arte se hicieron varios desarrollos del software, cuyas etapas serán descritas en el siguiente Capítulo.

Capítulo 5

Contribución

Partiendo de unas ideas preliminares resultado de analizar algunas series de detectives [2], hemos definido una familia de problemas característicos de videojuegos multijugador de detectives y hemos definido un modelo computacional para poder implementar participantes automáticos capaces de cumplir con los distintos roles del juego.

Los distintos cometidos que pueden adoptar nuestros personajes son los de testigos (los ratones), encargados de actuar en una primera fase y dialogar para aportar pruebas en una segudan, y el de detective (el científico), que se encarga de esclarecer los hechos y encontrar al culpable. Por lo tanto podemos distinguir tres partes principales:

1. Algoritmo BDI para la toma de decisiones de los ratones
2. Algoritmo dedicado a realizar preguntas por parte del científico y razonar con la información obtenida
3. Algoritmo dedicado a responder a las preguntas por parte de los ratones razonando qué respuesta es más beneficiosa para nuestros objetivos

A continuación describiremos en detalle cada una de la partes

5.1. Toma de decisiones de los ratones

La lógica dedicada al movimiento de los ratones está basada en las ideas principales de los agentes BDI descritos en el Capítulo 2. Siguiendo lo estudiado, nuestros ratones van a tener una serie de deseos que quieren satisfacer durante la ejecución.

Estos ratones perciben del entorno los cambios que se producen, es decir, las acciones de otros ratones sobre el laberinto. Es posible que un ratón no pueda percibir lo que ha hecho otro ratón debido a que no está en su campo de visión, pero sí que va a tener una noción de que el tiempo ha pasado para algún ratón y que posiblemente haya realizado alguna acción a pesar de no saber cuál.

El ciclo de ejecución que se espera en la ejecución para un tablero de $N \times M$ con 4 ratones es el siguiente:

1. Ratón1 ejecuta una acción.

   Ratón1, Ratón2, Ratón3, Ratón4 visualizan lo ocurrido

   - Si está dentro de su campo de visión recibe el evento y el estado del tablero
   - Si está fuera de su campo de visión recibe solo el tablero (la parte que ve)

2. Ratón2 ejecuta una acción.

   Ratón1, Ratón2, Ratón3, Ratón4 visualizan lo ocurrido

   - Si está dentro de su campo de visión recibe el evento y el estado del tablero
   - Si está fuera de su campo de visión recibe solo el tablero (la parte que ve)

3. Ratón3 ejecuta una acción.

   Ratón1, Ratón2, Ratón3, Ratón4 visualizan lo ocurrido

   - Si está dentro de su campo de visión recibe el evento y el estado del tablero
   - Si está fuera de su campo de visión recibe solo el tablero (la parte que ve)

4. Ratón4 ejecuta una acción.

   Ratón1, Ratón2, Ratón3, Ratón4 visualizan lo ocurrido

- Si está dentro de su campo de visión recibe el evento y el estado del tablero
- Si está fuera de su campo de visión recibe solo el tablero (la parte que ve)

Una vez finalizado el bucle reiteramos hasta que la noche termina (es decir se acaba el tiempo de movimiento de los ratones) y todos quedan en el último estado que tenían. El número de turnos de cada ratón está definido previamente y todos lo conocen.

La lógica debe ser consistente entre los diferentes turnos. No podemos querer comernos el queso en el primer turno y dejar de quererlo en el segundo sin que haya pasado nada relevante (como que otro ratón me haya visto).

Para determinar la acción los ratones deben recorrer sus deseos, los cuales están ordenados por prioridad, y realizar aquellos que sean más prioritarios. Un ratón intentará alcanzar tantos objetivos como pueda, siempre que los objetivos secundarios (aquellos que tienen una prioridad más baja) no entren en contradicción con los primarios (los que tienen una prioridad más alta). Entendemos que dos deseos están en contradicción cuando la consecución de uno de ellos implica la no consecución del otro.

Un ejemplo son los deseos de Descansar y Andar. Los ratones puedes querer descansar con una prioridad del 100 % y andar con un prioridad del 70 %. Si el ratón anda entonces no descansará, por lo tanto estamos incumpliendo un deseo de mayor prioridad. En este caso el ratón decidirá no hacer nada y cumplir su objetivo principal.

En cada iteración del ratón se van a realizar las siguientes acciones:

1. Actualizar la información del entorno con los eventos recibidos de las acciones del resto de ratones
2. Elegir que deseo es más relevante en el momento actual
3. Realizar una búsqueda de qué acciones debemos llevar a cabo para cumplir el objetivo elegido.

Dado que podemos perseguir elementos móviles y elementos fijos, en cada iteración podemos tener un tablero distinto al esperado en el que un deseo secundario se puede cumplir con muy poco esfuerzo mientras que otro más importante tiene un coste mayor. No se tiene un tiempo fijo para cumplir

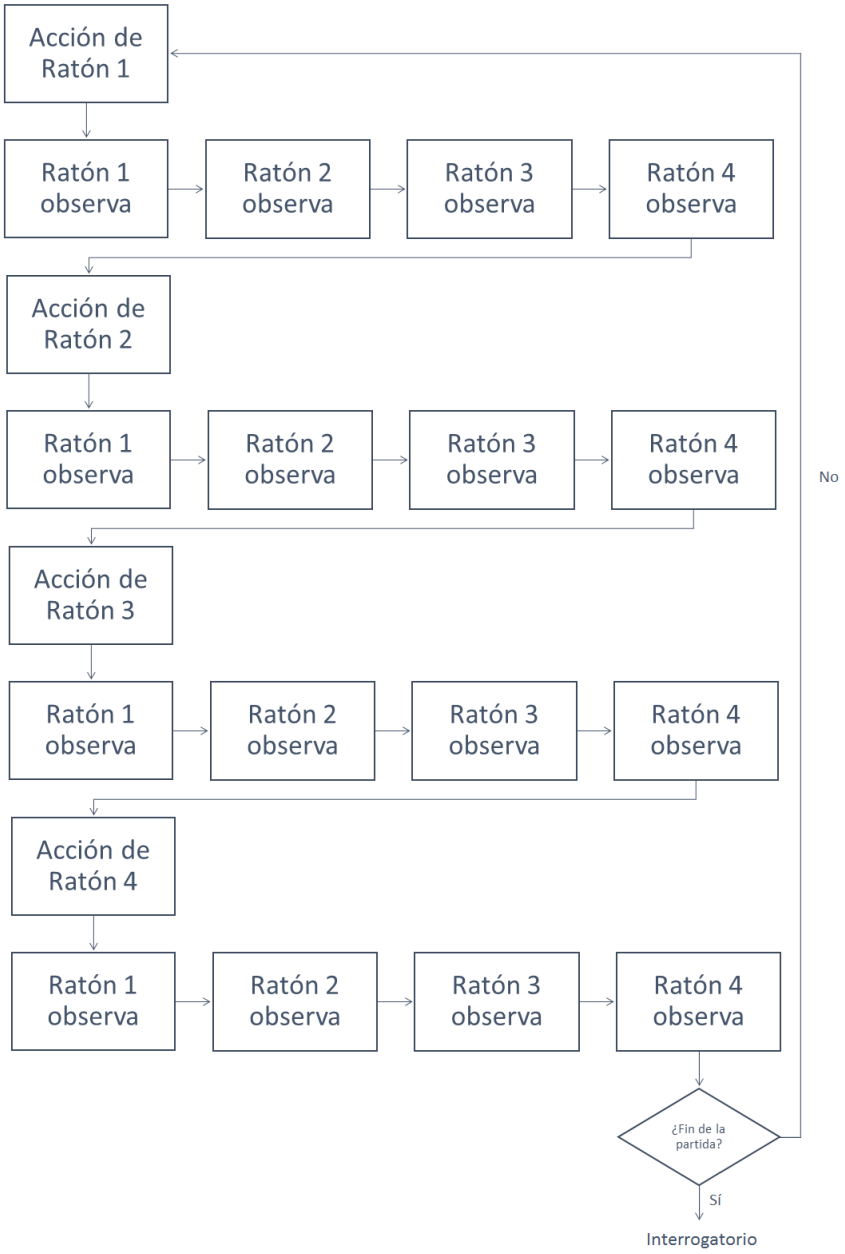

Figura 5.1: Diagrama de ejecución de los agentes

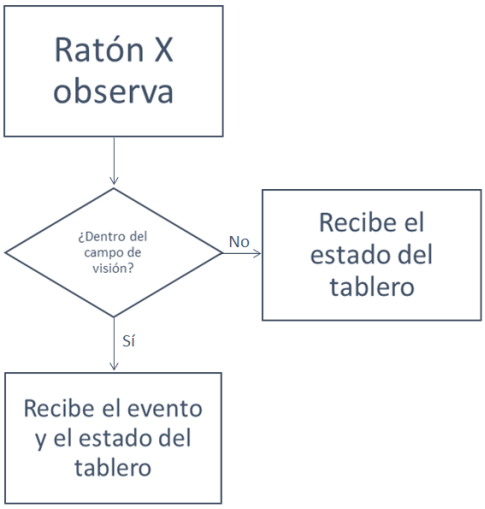

Figura 5.2: Diagrama de ejecución del evento Observar

los objetivos pero todos ellos están limitados por el número de turnos de la partida.

La elección de la acción a realizar va a consistir por tanto de los deseos y de su facilidad para alcanzarlos. En el caso de que queramos comernos el queso pero también deseamos visitar a otro ratón, haremos una ponderación entre la prioridad de ambos deseos y de lo cerca que estos están de nosotros. Tras esta ponderación elegimos el que más nos pueda beneficiar.

A la hora de implementarlo tenemos por tanto los siguientes elementos principales:

1. Deseos del ratón
2. Eventos que tienen lugar debido a las acciones de los ratones
3. Tablero y elementos relacionados con el entorno en el que se mueven
4. Algoritmos de decisión y búsqueda de los elementos

Veamos cada uno con más detalle

5.1.1. Deseos del ratón

Los ratones pueden tener los siguientes deseos:

- Comerse el queso
- No romper shojis
- Descansar
- Ver al ratón azul
- Ver al ratón rojo
- Ver al ratón verde
- Ver al ratón amarillo
- Caminar

Cada uno de los cuales tendrá un peso asociado que determina lo importante que es para el ratón ese deseo. Un ratón puede querer verse a sí mismo, razón por la que buscará un espejo.

5.1.2. Eventos

Los ratones pueden realizar las siguientes acciones:

- Comer. Solo un ratón puede comerse el queso, tras lo cual el queso desaparece
- Moverse al norte
- Moverse al sur
- Moverse al este
- Moverse al oeste
- Jugar (acción que se ejecuta cuando ve a otro ratón o un juguete). Un ratón puede jugar con varios ratones a la vez si están todos en la misma casilla.
- Esperar

Un evento contiene información de la acción ejecutada, el ratón que la llevó a cabo, si tuvo éxito o no y en qué momento tuvo lugar. Dado que los ratones pueden razonar y deducir eventos estos pueden ser más o menos fiables en función de cómo los hayamos obtenido.

5.1.3. Tablero y elementos relacionados

La interfaz del tablero permite obtener información de las casillas que hay a nuestro alrededor. Las casillas pueden ser:

- Bloques que obstaculizan el paso
- Shojis, los cuales son un tipo de casilla especial que se puede romper transformandola en otro tipo de casilla (shoji roto).
- Shojis rotos. Son el resultado de una casilla tipo shoji por la que ha pasado un ratón.
- Transitables. Las casillas transitables pueden contener elementos. Estos elementos son:
  - El queso
  - Los ratones
  - Juguetes
- Desconocidas. No son una casilla como tal, lo que ocurre es que el ratón no puede visualizar esta casilla y toman este valor. Cuando la casilla entra en su campo de visión pasa a tener alguno de los valores anteriores.

Todos los elementos de las interfaces permiten obtener información del lugar en el que nos encontramos.

5.1.4. Algoritmos de decisión y búsqueda de elementos

El algoritmo más importante en el movimiento es el que decide la acción que debemos realizar en función de los deseos. Este algoritmo revisa los deseos del ratón por orden de prioridad e intenta cumplir todos aquellos que no entren en contradicción con otro más prioritario.

Los deseos que puede tener un ratón son:

- Comerse el queso
- Ver a otro ratón o a sí mismo
- Caminar
- No romper shojis
- Terminar la partida en la casilla inicial
- Descansar

Los deseos de comerse el queso y ver a otro ratón implican movimiento, por lo tanto si intentamos satisfacer uno de los dos, al mismo tiempo estamos cumpliendo el de caminar. No romper shojis y terminar en la casilla inicial, determinan qué camino elige el ratón para moverse por el tablero. En general busca el camino más corto, pero si esto implica romper shojis, puede que tengamos que buscar un camino alternativo más largo en caso de que no romper shojis forme parte de sus deseos.

Terminar en la casilla de salida implica que a medida que se acerca el final de la partida los ratones cambien sus objetivos por el de volver a sus casillas iniciales. Puede cumplirse a la vez que se cumplen los deseos de comerse el queso, ver a otro ratón o caminar, aunque si el límite de la partida está muy cerca, tomará más prioridad volver.

El deseo de descansar entra en contradicción con todos los deseos que implican movimiento, por lo tanto solo se ejecutará una posible acción: esperar.

Una vez que se conoce el deseo que queremos alcanzar exploramos las distintas intenciones que puede tener el ratón. Estas intenciones nos dirán si debemos movernos o debemos permanecer en el sitio y ejecutar la acción que nos permite cumplir el deseo. Para el movimiento y el cálculo de rutas se utiliza el algoritmo A^*.

5.2. Mente del científico: preguntas

El científico se va a dormir conociendo el estado inicial del laberinto. Sabe dónde están los ratones, el queso y dónde había shojis. Cuando se despierta ve el nuevo estado del laberinto, con las posibles nuevas posiciones de los

Figura 5.3: Relaciones de los deseos

ratones, shojis rotos y otros enteros y el queso puede haber desaparecido.

Tras comprobar que el queso no está comienza el interrogatorio hacia los ratones. Podría ser que en la ejecución ningún ratón se comió el queso y por lo tanto no se tiene que castigar a nadie, sin embargo esta ejecución carece de interés para nuestro juego donde queremos poner a prueba la inteligencia del científico y de los ratones a la hora de formular y contestar preguntas respectivamente.

Aquellos experimentos en los que ningún ratón se come el queso se consideran fallidos por lo que se debería realizar una nueva ejecución con distintos datos de entrada. Si en un laberinto es imposible que ningún ratón se coma el queso lo llamaremos laberinto inválido.

El científico debe determinar quien se ha comido el queso, pero lo más probable es que ningún ratón se declare culpable por sí mismo. Debido a esto tiene que saber que ratones han estado más cerca del queso y cuáles tienen coartadas más creíbles para sacar conclusiones sobre quién fue el responsable. El científico actuará de la siguiente manera:

En primer lugar el científico preguntará uno por uno a ver quien se ha comido el queso.

Si uno solo responde afirmativamente y el resto negativamente, este será

el principal sospechoso (pero no necesariamente el culpable, razón por la que el interrogatorio continuará). Se considera que los ratones pueden ser poco astutos o incluso compincharse unos con otros para salvarse entre sí (un ejemplo real es el caso de Marta del Castillo).

Posteriormente se buscará profundizar sobre la cercanía de un ratón al queso. Para ello se observará los shojis que se han roto por la noche y a los ratones más cercanos a esos shojis se le preguntará si estuvieron ahí.

En caso de que un ratón afirme haber roto ese shoji, se tiene una pista de la cercanía del ratón al queso. Si ningún ratón afirma haber estado allí, se va a buscar que ratones tienen dicho shoji en su camino al queso. Los que puedan haber roto ese shoji serán acusados de mentirosos y penalizados como tal. Si ningún ratón tiene que pasar por ese shoji vamos a considerar que ha pasado el que tiene la casa más cerca del shoji, pero no le aplicamos penalización (puede que no haya sido él realmente).

En este caso no consideramos que los ratones nos intentan engañar con el shoji, y por lo tanto tampoco se tienen en cuenta ratones estúpidos que dicen que han estado ahí sin estar.

Cuando sabemos las posiciones más cercanas de cada ratón al queso, vamos a hacer una ponderación de la confianza en cada ratón y de su cercanía al queso. Aquel que obtenga una menor credibilidad de no haberse comido el queso será acusado culpable.

Antes de condenar definitivamente a un ratón, vamos a preguntar a cada uno si otro ratón se ha comido el queso. Este otro ratón por el que vamos a preguntar es el que nosotros consideramos culpable, siempre que no le estemos preguntando a un ratón si efectivamente fue él quien se comió el queso (algo que ya hemos hecho antes). El principal sospechoso va a ser preguntado por nuestro segundo sospechoso. Con esto modificaremos la confianza que tenemos en cada ratón pudiendo alterarse el resultado.

Finalmente se dará el veredicto sobre qué ratón condenamos.

Esquemáticamente, la mente del científico funciona de la siguiente manera:

1. El científico pregunta a cada ratón si se han comido el queso

   a) Uno responde afirmativamente y el resto negativamente: tenemos un sospechoso principal
   b) Todos responde negativamente: no tenemos ningún sospechoso
   c) Más de uno responde afirmativamente: tenemos varios sospechosos principales, pero reducimos la confianza de los que han dicho que sí, porque sabemos que al menos uno nos ha mentido

2. El científico pregunta a cada ratón si ha pasado por un cierto shoji
   a) Al menos un ratón responde afirmativamente: tenemos posiciones de proximidad al queso por parte de algunos ratones.
   b) Ningún ratón afirma haber roto el shoji: calculamos las rutas de la posición inicial al queso
      1) algún ratón pasa por ese shoji: reducimos la confianza por mentirnos y tenemos la posición de los ratones
      2) ningún ratón debe pasar por ese shoji: acusamos al que está más cerca de su casilla inicial pero mantenemos la confianza de ese ratón (puede ser inocente de esta acusación)

3. El científico pregunta a cada ratón si su principal sospechoso es quien se comió el queso.
   a) Los ratones confirman que este ratón se comió el queso: se refuerza esta teoría
   b) Los ratones niegan que este ratón se comió el queso: se debilita la teoría y se reduce la confianza en el ratón que respondió.

4. El científico acusa a los sospechosos en función de su cercanía al queso durante la ejecución y de las respuestas previas.
   a) Solo uno de los sospechosos se declara culpable: es condenado dicho ratón
   b) Ninguno o más de uno se declara culpable: es condenado aquel que tiene una menor confianza

El científico puede por lo tanto realizar dos tipos de preguntas:

- Si un cierto ratón se ha comido el queso
- Si un cierto ratón estuvo en la casilla (X,Y)

Se admite que el científico le pregunte a un ratón por el mismo o por otros.

El científico usa una interfaz para comunicarse con la IA de los ratones. Esta interfaz además de contener los métodos necesarios para poder realizar las preguntas, contiene la información obtenida durante el interrogatorio. Entre los datos más relevantes que se guardan están:

- La confianza en el ratón
- Un array con las casillas visitadas
- Un booleano indicando si en algún momento se ha inculpado o ha sido inculpado por alguien

Además, esta clase es la encargada de determinar la acusación que se le da al ratón. La acusación final por parte del científico es un valor entre cero y cien que indica el porcentaje de culpabilidad que, según el científico, tiene el ratón.

Será elegido aquel ratón cuya culpabilidad sea mayor.

5.3. Mente de los ratones: respuestas

Los ratones además del movimiento, tienen que razonas para responder al científico. Las respuestas que el ratón puede dar son las siguientes:

- Si
- No
- Quizás
- No lo sé
- No responder/Silencio

A partir de las posibles respuestas se han realizado "mentes" para los ratones muy simples que siempre responden lo mismo. Las tres estas mentes son:

- Siempre responder sí

- Siempre responder no
- Siempre responder no lo se

Estas soluciones son muy básicas y no tienen razonamiento sobre los hechos. La finalidad de estas mentes es simplemente realizar pruebas forzando ciertas respuestas.

La solución más sencilla donde realmente se utiliza el conocimiento recopilado es en la IA que podemos llamar "ratón honesto". Este ratón siempre dice la verdad a cada pregunta. Para ello, cada vez que se le pregunta sobre si alguien se ha comido el queso o alguien ha estado en un cierto lugar busca en su memoria si esto ha ocurrido. Si tiene constancia del suceso, responderá con si o no. En caso de no tener datos sobre lo que se le pregunta responderá con no lo sé.

Puede ser útil tener ratones que digan la verdad para hacer distintas pruebas, pero este no era el objetivo que perseguíamos: crear ratones capaces de mentir de forma creíble y engañar al científico que les interroga para que no les inculpen. Debido a esto existe una quinta inteligencia, el "ratón inteligente", el cual pretende desviar lo máximo posible al científico de su proximidad al queso.

Se han desarrollado dos versiones del ratón inteligente. La primera versión es un poco más sincera y por lo tanto menos efectiva. Difiere del ratón sincero en que responde que no cuando le preguntan si él se ha comido el queso. Esta versión solo inculpará a otro ratón de haberlo hecho si lo ha visto comiéndose el queso. Cuando le pregunten por las casillas que ha recorrido, siempre evitará afirmar que estuvo en aquellas casillas que están en su trayectoria hacia el queso, ya que podrían dar pistas al científico de que se intentó comer el queso.

La segunda versión del ratón inteligente deduce posibles culpables y les intentará echar la culpa de acciones que les sitúe cerca de la escena del crimen. Para ello, una vez finalizada la ejecución el ratón inteligente tiene la opción de sacar conclusiones. Cada ratón va a revisar su conocimiento en búsqueda de las distintas posiciones de los ratones. Si otro ratón tuvo éxito en comerse el queso tenemos un culpable. Si ninguno se comió el queso están a salvo, aunque como se indicó anteriormente estos experimentos se consideran fallidos e irrelevantes. Si el ratón se ha comido el queso o ha sido otro ratón pero no lo ha visto, debe buscar alguien a quien culpar. Ese alguien va a ser el ratón

que más cerca haya pasado del queso. Se elige a este ratón porque, al estar cerca de la escena del crimen, puede que otros ratones también la incriminen.

Una vez que el ratón tiene un sospechoso, deberá utilizar esta información convenientemente para responder cuando le pregunten sobre quién se comió el queso. Si a un ratón le preguntan si el se comió el queso siempre responderá que no (aunque haya sido él no se va a inculpar).

Si le preguntan por otro ratón y él se ha comido el queso intentará desviar la atención hacia otro posible culpable. Por eso si es preguntado por el ratón que considera que es un sospechoso importante responderá que sí, pero si le preguntan por cualquier otro ratón responderá que no lo sabe. En caso de que no se haya comido el queso pero tiene altas probabilidades de ser inculpado de nuevo intentará desviar la atención a otro ratón del mismo modo. Solo en caso de que las pruebas no le incriminen responderá de manera más sincera. En este caso dirá que sí cuando ha visto que el ratón por el que preguntan se ha comido el queso. En caso de que no lo haya visto pero crea que es el ratón que probablemente se lo haya comido responderá que quizás. Si le preguntan por cualquier otro ratón responderá que no.

Si le preguntan por las casillas que el ratón recorrió, responderá que no en caso de que estas casillas estén en la ruta del ratón al queso, en posiciones cercanas a este último o cuando realmente no haya estado en esa casilla. Responderá que sí cuando las casillas le sitúen cerca de casa y no responderá cuando le pregunten por posiciones lejanas a su casilla inicial en las que verdaderamente ha estado.

Dado que los términos cerca y lejos son demasiado ambiguos se toma como punto de partida que el queso está lejos, para así evitar situar al ratón en posiciones que den a entender que se ha podido comer el queso. Las posiciones cercanas son aquellas que están a la mitad de distancia o menos que el queso. Cuando le preguntan si otro ratón estuvo en una casilla responderá sinceramente en base al conocimiento que ha obtenido durante la ejecución.

El ratón puede no percibir algunos eventos que ocurren a su alrededor debido a que no están dentro de su campo de visión. Para completar esta falta de información se decidió construir un sistema de reglas similar a ***Jess***. Las reglas completan la información acerca de los posibles sucesos que han tenido lugar mientras no veíamos.

5.4. Herramientas utilizadas en el desarrollo

El lenguaje de programación que hemos utilizado para desarrollar al algoritmo BDI encargado de las acciones de los ratones y el sistema de conversación y razonamiento tanto del científico como de los ratones es Java. Aunque no se ha utilizado, Jess sirvió como guía para construir los procedimientos dedicados a completar el conocimiento de forma similar a un sistema de reglas. El alojamiento del código y el control de versiones se ha realizado mediante la plataforma GitHub.

5.4.1. Java

Java es una lenguage de programación de proposito general, concurrente y orientado a objetos [19]. Se trata de un lenguaje multiplataforma, de manera que los desarrolladores pueden escribir el código desde diferentes dispositivos. El código es precompilado y ejecutado en una máquina virtual (JVM) independientemente de la arquitectura del computador.

Esta relacionado con los lenguajes C y C++, aunque difiere de ambos en varios aspectos. No tiene tantas utilidades de bajo nivel e incluye algunas ideas de otros lenguajes de programación. Se trata de una lenguaje fuertemente tipado, produciendo errores en tiempo de compilación.

5.4.2. Jess, the Rule Engine for the Java Platform

Jess es un lenguaje principalmente declarativo que permite construir sistemas basados en reglas utilizando Java [21]. Se trata de uno de los lenguajes para construir sistemas basados en reglas más potente que existen. Este lenguaje permite construir un tipo de inteligencia artificial que recibe el nombre de sistemas expertos. Un sistema experto es una colección de reglas que se aplican repetidamente a un conjunto de hechos que representan el conocimiento del mundo. Recientemente se ha utilizado también para desarrollar agentes inteligentes.

Jess utiliza una versión mejorada del algoritmo **RETE** para el procesado de reglas. RETE es un mecanismo que resuelve el problema de encaje de patrones de manera mucho más eficiente en una cadena de condiciones *if* dentro de un bucle. Jess además cuenta con mejoras como el encadenamiento

hacia atrás.

Jess, que en un principio surgió como una copia de **CLIPS** para Java, cuenta con una ventaja fundamental: permite crear y manipular clases de Java directamente. Además permite llamar a métodos de Java o implementar interfaces de este lenguaje. Estas razones han conseguido que se distinga por méritos propios de su precedente CLIPS.

5.4.3. GitHub

GitHub es un repositorio web de tipo Git. Ofrece la funcionalidad típica del control de versiones distribuido y gestión del código de un sistema Git además de proporcionar funcionalidades adicionales propias. GitHub cuenta con una interfaz web y otra de escritorio además de los comandos para la consola con los que ya cuenta Git.

El código implementado se puede encontrar en el siguiente enlace:

`https://github.com/alexvillarin/MouseMind`

Capítulo 6

Resultados y discusión

6.1. Pruebas con el movimiento de los ratones

Lo primero que se implementó fue el movimiento de los ratones. Una vez acabado se hizo una prueba con el laberinto que aparece en la siguiente figura:

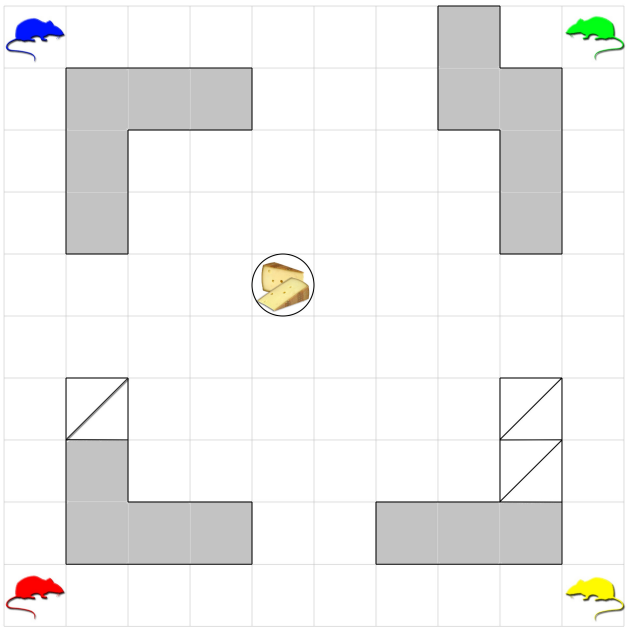

Figura 6.1: Estado inicial del tablero

(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)	(1,8)	(1,9)	(1,10)
(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)	(2,8)	(2,9)	(2,10)
(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)	(3,8)	(3,9)	(3,10)
(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)	(4,8)	(4,9)	(4,10)
(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)	(5,8)	(5,9)	(5,10)
(6,1)	(6,2)	(6,3)	(6,4)	(6,5)	(6,6)	(6,7)	(6,8)	(6,9)	(6,10)
(7,1)	(7,2)	(7,3)	(7,4)	(7,5)	(7,6)	(7,7)	(7,8)	(7,9)	(7,10)
(8,1)	(8,2)	(8,3)	(8,4)	(8,5)	(8,6)	(8,7)	(8,8)	(8,9)	(8,10)
(9,1)	(9,2)	(9,3)	(9,4)	(9,5)	(9,6)	(9,7)	(9,8)	(9,9)	(9,10)
(10,1)	(10,2)	(10,3)	(10,4)	(10,5)	(10,6)	(10,7)	(10,8)	(10,9)	(10,10)

Figura 6.2: Valores de las casillas

A cada ratón se le asignó una serie de deseos para que comenzaran a moverse y realizar acciones.

1. Azul
    a) Comerse el queso - 100
    b) No romper shojis - 70
2. Verde
    a) Caminar - 80

b) Comerse el queso - 70

c) No romper shojis - 70

3. Rojo

   a) Comerse el queso - 100

   b) No romper shojis - 70

4. Amarillo

   a) Descansar - 100

   b) No romper shojis - 70

La ejecución duró un total de 20 turnos para cada ratón.

Los resultados destacables de la ejecución fueron los siguientes:

- El ratón azul se come el queso porque es el que está más cerca
- El ratón azul no se mueve una vez que se ha comido el queso porque así satisface todos sus deseos restantes. Su posición final es la (5,6)
- El ratón verde intenta comerse el queso. Como no llega antes que el azul abandona su intento de comerse el queso y se dedica a caminar. Su posición final es la (1,3)
- El ratón rojo intenta comerse el queso. Como no llega antes que el azul abandona su intento y descansa, ya que así satisface todos sus deseos restantes. Su posición final es la (7,6).
- El ratón amarillo descansa durante toda la ejecución. Su posición final es la (10,10)

Tras estos resultados se puede observar que hay algo que falla y es que los ratones, una vez que alguien se come el queso, si no desean seguir caminando, se quedan como estatuas en la posición que tuvieran. Esto es algo fatal, ya que el ratón que se comió el queso se queda en la misma posición del queso, y el resto, que no llegaron a poder comerlo, en posiciones muy cercanas a él.

Esto nos llevó a añadir un nuevo deseo: Evitar ser castigado. Este deseo hace que un ratón quiera volver a casa para no levantar sospechas sobre lo que hizo por la noche.

Utilizando el mismo laberinto y el mismo número de turnos realizamos una nueva ejecución, ahora con los siguientes deseos:

1. Azul
    a) Comerse el queso - 100
    b) No romper shojis - 70
    c) Evitar ser castigado - 10
2. Verde
    a) Caminar - 80
    b) Comerse el queso - 70
    c) No romper shojis - 70
    d) Evitar ser castigado - 10
3. Rojo
    a) Comerse el queso - 100
    b) No romper shojis - 70
    c) Evitar ser castigado - 10
4. Amarillo
    a) Descansar - 100
    b) No romper shojis - 70
    c) Evitar ser castigado - 10

Los resultados de la ejecución variaron bastante respecto a la ejecución anterior y fueron más acordes a un ratón inteligente que evita ser castigado.

- El ratón azul se come el queso porque es el que está más cerca
- El ratón azul vuelve a su posición inicial, evitando romper shojis, para no ser castigado. Su posición final es la (1,1)
- El ratón verde intenta comerse el queso. Como no llega antes que el azul abandona su intento de comerse el queso y se dedica a caminar. Su posición final es la (7,10)
- El ratón rojo intenta comerse el queso. Como no llega antes que el azul abandona su intento e intenta volver a casa para no ser castigado. En su camino de vuelta evita romper shojis. Su posición final es la (10,1).

- El ratón amarillo descansa durante toda la ejecución. Su posición final es la (10,10)

Dado que en las anteriores ejecuciones no habíamos comprobado el deseo de querer ver a otro ratón, hicimos una nueva ejecución añadiendo este deseo.

Utilizando el mismo laberinto y el mismo número de turnos realizamos una nueva ejecución, ahora con los siguientes deseos:

1. Azul
    a) Comerse el queso - 100
    b) No romper shojis - 70
    c) Evitar ser castigado - 10
2. Verde
    a) Caminar - 80
    b) Ver al ratón azul - 70
    c) No romper shojis - 70
    d) Evitar ser castigado - 10
3. Rojo
    a) Comerse el queso - 100
    b) No romper shojis - 70
    c) Evitar ser castigado - 10
4. Amarillo
    a) Descansar - 100
    b) No romper shojis - 70
    c) Evitar ser castigado - 10

Esta vez los resultados no fueron muy distintos a los de la ejecución anterior, pero podemos destacar que esta vez el ratón verde se paró a jugar con el ratón azul en el turno número 10, que coincide justamente cuando el ratón azul estaba comiéndose el queso. Tras esto, el ratón azul volvió a su posición inicial y el ratón verde se dedicó a caminar.

Con esto comprobamos que el algoritmo funcionaba correctamente para todos los deseos y que estos se cumplían de una forma lógica. Sin embargo, todavía se podía explorar mucho más acerca de las acciones.

Se podría añadir una variante a los deseos, haciendo que estos no sean atómicos. Si un ratón quiere jugar con otro puede que sea muy juguetón, y no tenga suficiente con jugar una única vez, sino que quiera hacerlo en repetidas ocasiones. Para llevarlo a cabo tendríamos varias opciones. La primera y la que parece más lógica es añadir un campo de completitud al deseo que indique cuando está satisfecho. Sin embargo, podría ser interesante que el propio peso del deseo actúe como indicador de si se ha cumplido o no. Según esto, conforme un deseo se está satisfaciendo, bajaría su prioridad. Cuando la prioridad del deseo es cero, entendemos que se ha completado y se elimina.

Según lo anterior, el deseo de comerse el queso podría ser satisfecho comiendolo en varias ocasiones. Pero en este caso hay un problema añadido, el queso puede acabarse, por lo que el deseo no se acabaría de satisfacer. En esta modalidad, podría haber más de un queso y se juega con más variables, ya que los quesos se los puede haber comido un único ratón o varios.

6.2. Pruebas con la mente del científico y de los ratones

Se realizó una primera prueba ejecutando el movimiento de los ratones seguido de un interrogatorio por parte del científico. El tablero era el mismo que en la Figura 6.1, se realizó una partida con 20 turnos y los deseos eran de nuevo los mismo que en la segunda prueba:

1. Azul
    a) Comerse el queso - 100
    b) No romper shojis - 70
    c) Evitar ser castigado - 10
2. Verde
    a) Caminar - 80
    b) Comerse el queso - 70
    c) No romper shojis - 70

d) Evitar ser castigado - 10

3. Rojo

   a) Comerse el queso - 100

   b) No romper shojis - 70

   c) Evitar ser castigado - 10

4. Amarillo

   a) Descansar - 100

   b) No romper shojis - 70

   c) Evitar ser castigado - 10

En esta primera prueba todos los ratones utilizaron como IA para las respuestas aquella que dice siempre la verdad, por lo tanto cuando el científico preguntó a los ratones si se habían comido el queso, el que efectivamente lo hizo se delató. Evidentemente ante la falta de más pruebas, el científico acusó al ratón azul que es el que se comió el queso.

Los resultados destacables de esta ejecución fueron:

- El científico preguntó a cada ratón si se había comido el queso. Todos respondieron que no excepto el azul, que se lo había comido y se delató.
- No se rompió ningún shoji durante la ejecución, por lo que el científico no pregunto a ningún ratón acerca de sus posiciones.

En la segunda prueba los ratones utilizaron como IA para las respuestas la que llamamos ratón inteligente. Esta IA mentía para desviar la atención de ellos si se habían comido el queso. Cuando el científico les preguntó si se habían comido el queso respondieron que no y cuando les preguntó por haber roto un shoji, solo dijeron la verdad si no estaba cerca de su ruta al queso. En este caso el científico acusó al ratón verde, y por lo tanto falló en su juicio. Esto es debido a que ante la falta de pruebas, el científico decidió acusar al ratón porque no volvió a su posición inicial.

De esta ejecución podemos destacar los siguientes resultados:

- El científico preguntó a cada ratón si se había comido el queso. Todos respondieron que no, por lo tanto el científico no obtuvo ninguna pista de estas preguntas.

- No se rompió ningún shoji durante la ejecución, por lo que el científico no pregunto a ningún ratón acerca de sus posiciones.
- Todos los ratones estaban en su posición inicial excepto el verde, lo cual daba información de que se había movido.

Posteriormente cambiamos el laberinto y forzamos al ratón azul a que tuviera que romper al menos un shoji para llegar al queso, lo que hizo cambiar las acusaciones.

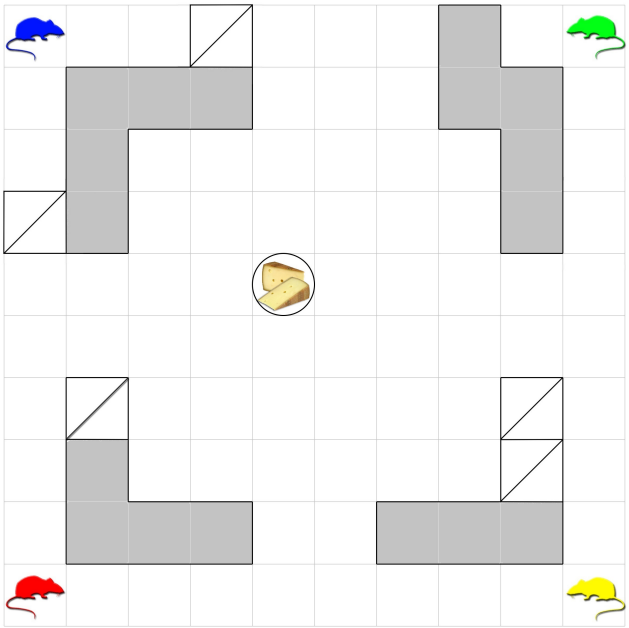

Figura 6.3: Estado inicial del tablero en el segundo interrogatorio

En el laberinto de la Figura 6.3 el ratón azul está obligado a romper los shojis si quiere comerse el queso. Esto provoca que el científico sospeche de él y lo acuse de haberse comido el queso.

Los resultados más relevantes de esta ejecución fueron:

- El ratón azul rompió el shoji de la casilla (1,4) y se comió el queso. El ratón verde y el rojo también se lo intentaron comer, pero al fracasar volvieron a sus casillas iniciales. El ratón amarillo no se movió.

- El científico pregunta a todos los ratones si se comieron el queso. Todos responden que no.
- El científico pregunta a todos los ratones si estuvieron en la casilla (1,4) porque hay un shoji roto. Todos responden que no pero cree que el azul si que estuvo porque es el más cercano.
- El científico decide acusar al ratón azul. Pregunta al ratón azul si el amarillo se comió el queso. Este responde que no lo sabe porque su principal sospechoso es el verde y no el amarillo. Luego pregunta a todos los ratones si el azul se comió el queso. Todos responden que si corroborando la versión que tenía.
- El científico castiga al ratón azul y acierta

Este es el resultado que más información aporta, por lo que se detalla completo en el Anexo E: Ejemplo de ejecución completo

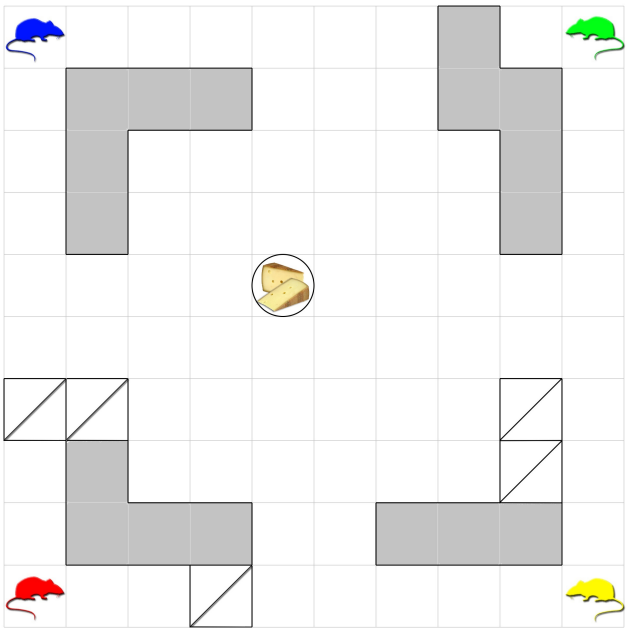

Figura 6.4: Estado inicial del tablero en el tercer interrogatorio

Un caso similar ocurre con el ratón rojo en el laberinto de la Figura 6.4. Al romper los shojis para intentar comerse el queso el científico lo acusa, falsamente, de ser el culpable del crimen. En este caso el ratón azul ha conseguido

engañar al científico. Los resultados sin embargo son ligeramente distintos, ya que los ratones responden que no cuando el científico pregunta si el rojo es el culpable de comerse el queso. A pesar de ello lo acusa por falta de pruebas contra el verdadero culpable, el ratón azul.

Capítulo 7

Conclusiones

A pesar de que la industria de los videojuegos se esfuerza en mejorar, a día de hoy sigue careciendo de un método eficiente para servir a los personajes de una inteligencia con la que mantener conversaciones lógicas en base al conocimiento obtenido.

Se han logrado grandes avances relacionados con el diseño gráfico y de la propia historia. Una prueba de esto son los complejos y variados equipos dedicados a desarrollar un videojuego. Muy distinto es el caso del manejo de la información, razón por la cual puede ser el próximo paso que realmente suponga una verdadera revolución en la industria.

Para mejorar en este campo se deben estudiar nuevas técnicas de deducción y razonamiento en base a los descubrimientos realizados en el campo del entendimiento humano. Las dos características que más pueden ayudar en este sentido son la capacidad de mentir y los problemas relacionados con el olvido.

Crear personajes que entiendan lo que ha ocurrido y saquen conclusiones sobre qué pueden pensar los demás de eso y de qué forma se puede variar los acontecimientos para que me favorezcan es el punto principal.

Por otra parte, añadir a nuestros personajes la característica de poder olvidar un cierto conocimiento obtenido por no ser bastante relevante proporciona realismo a la historia.

Los personajes tendrán que ser capaces de simular cómo va a reaccionar una persona frente a estos hechos. Queremos evitar caer de nuevo en el error de utilizar resultados fijos, por lo tanto la forma de razonar debe

estar influida por los hechos que han ocurrido. Tanto el tiempo como el lugar en el que ocurrió una acción deben tenerse en cuenta para sacar conclusiones. Esto junto a la posibilidad de que el conocimiento se deteriore o que sea impreciso da lugar a resultados de incertidumbre más cercanos a un caso real.

Estas ideas se han tenido en cuenta a la hora de realizar este proyecto. Hemos definido un sistema de agentes inteligentes que son capaces de explorar su entorno y tomar decisiones acordes a sus percepciones, de forma que puedan lograr un cierto objetivo.

Posteriormente estos agentes virtuales utilizan ese conocimiento para responder a una serie de preguntas. Este conocimiento es utilizado de manera consciente, logrando mentir para evitar un castigo.

Todo esto se ha formalizado en un modelo que debe cumplir las soluciones a esta familia de problemas. El algoritmo encargado de las acciones de los ratones que tiene lugar durante la noche extrae las ideas fundamentales de los agentes inteligentes para poder conocer el entorno. El razonador permite obtener conclusiones que a priori no se tienen que conocer proporcionando la capacidad de conversar sobre un mayor dominio.

Los resultados de varias ejecuciones demuestran que estos ratones son capaces de buscar y encontrar sus objetivos por el tablero a pesar de que estos puedan estar cambiando constantemente. Cuando los ratones persiguen el queso pero este desaparece cambian de objetivo y empiezan a realizar otra acción para satisfacer el resto de sus deseos. Hemos conseguido lograr ratones con un comportamiento más realista.

También podemos afirmar que utilizar las mentiras proporciona mayor interés a la trama. Los ratones consiguen ocultar sus acciones utilizando mentiras o respuestas ambiguas que no les delatan. Los porcentajes tan parecidos demuestran que el científico duda a la hora de acusar a un cierto ratón. Nuestros agentes elijen las respuestas de forma consciente y con un determinado fin.

El trabajo realizado presenta de manera precisa los problemas descritos, los cuales estaban presentes en muchos videojuegos pero no se habían planteado como tales. Esto ofrece oportunidades para futuras líneas de investigación relacionadas con este tema y presenta una base para profundizar en este campo.

Anexo A

Resolutor de problemas STRIPS

STRIPS es un resolutor de problemas que trata de encontrar una secuencia de acciones dentro de un espacio de posibles acciones de manera que nos permitan transformar el estado inicial del mundo en un estado final que satisface unas ciertas condiciones. STRIPS representa el mundo como un conjunto de predicados y puede trabajar con una gran cantidad de fórmulas. Una instancia de STRIPS se compone de:

1. Un estado inicial
2. La especificación de estados de meta
3. Un conjunto de acciones, para cada una de las cuales se incluyen:
    - Precondiciones
    - Postcondiciones

Matemáticamente, una instancia de STRIPS es una tupla $\langle P, O, I, G\rangle$, donde:

- $\boldsymbol{P}$ es un conjunto de condiciones (es decir, variables proposicionales).
- $\boldsymbol{O}$ es un conjunto de operadores (es decir, acciones); cada operador es también una tupla $\langle \alpha, \beta, \gamma, \delta\rangle$, donde cada elemento es un conjunto de condiciones:
    - α representa a las condiciones que deben ser verdaderas para que la acción pueda ejecutarse.

- β representa a las condiciones que deben ser falsas para que la acción pueda ejecutarse.
- γ representa a las condiciones se hacen verdaderas si se ejecuta la acción.
- δ representa a las condiciones se hacen falsas si se ejecuta la acción.

- $\boldsymbol{I}$ es el estado inicial, dado por un conjunto de condiciones que son inicialmente verdaderas (todas las demás se asumen falsas).

- $\boldsymbol{G}$ es la especificación del estado de meta, que está dada por una dupla $\langle N, M\rangle$, que especifica qué condiciones deben ser verdaderas y cuáles falsas para considerar a un estado como meta.

Un plan para una instancia es una secuencia de operadores que puede ser ejecutada desde el estado inicial, y que lleva hasta un estado meta.

Formalmente, un estado es un conjunto de condiciones, y se representa por el conjunto de condiciones que son verdaderas en él. Las transiciones entre estados se modelan mediante una función de transición, que es una función que mapea estados en otros estados que resultan de aplicarles acciones a los primeros. Ya que los estados se representan por conjuntos de acciones, la función de transición emparentada con la instancia STRIPS $\langle P, O, I, G\rangle$ es una función:

$$\text{sucesor} : 2^P \times A \to 2^P$$

donde 2^P es el conjunto de todos los subconjuntos de P, y por lo tanto es el conjunto de todos los posibles estados.

La función de transición puede definirse, asumiendo que las acciones siempre pueden ser ejecutadas pero no tienen efecto si sus precondiciones no se cumplen, como:

$$\text{sucesor}(C, \langle \alpha, \beta, \gamma, \delta\rangle) = \begin{cases} (C \backslash \delta) \cup \gamma & \text{Si } (\alpha \subseteq C / \beta \cap C = \emptyset) \\ P & \text{De otra forma} \end{cases}$$

La función sucesor puede extenderse para secuencias de acciones mediante ecuaciones recursivas:

$$\text{sucesor}(C, [\,]) = C$$
$$\text{sucesor}(C, [a_1, a_2, \ldots, a_n]) = \text{sucesor}(\text{sucesor}(C, a_1), [a_2, \ldots, a_n])$$

Un plan para una instancia de STRIPS es una secuencia de acciones cuya ejecución ordenada produce un estado que satisface las condiciones de meta, a partir del estado inicial.

Formalmente, $[a_1, a_2, \ldots, a_n]$ es un plan para $G = \langle N, M \rangle$ si $F = \text{sucesor}(I, [a_1, a_2, \ldots, a_n])$ satisface:

$$N \subseteq FN \subseteq F \qquad\qquad M \cap F = \emptyset M \cap F = \emptyset$$

Anexo B

Distribución Beta

En teoría de la probabilidad y estadística, la función de distribución beta es una familia de funciones de distribución continuas definida en el intervalor $[0, 1]$ y parametrizada por dos valores positivos (α y β).

La función de densidad de la distribución beta es:

$$f(x) = \begin{cases} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1} & \text{si } 0 < x < 1 \\ 0 & \text{en caso contrario} \end{cases}$$

donde Γ es la función gamma:

$$\Gamma(z) = \int_0^\infty t^z e^{-t} dt$$

Función de distribución

$$F(x) = \begin{cases} 0 & \text{si } x \leq 0 \\ \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \int_0^x y^{\alpha-1}(1-y)^{\beta-1} dy & \text{si } 0 < x < 1 \\ 1 & \text{si } x \geq 1 \end{cases}$$

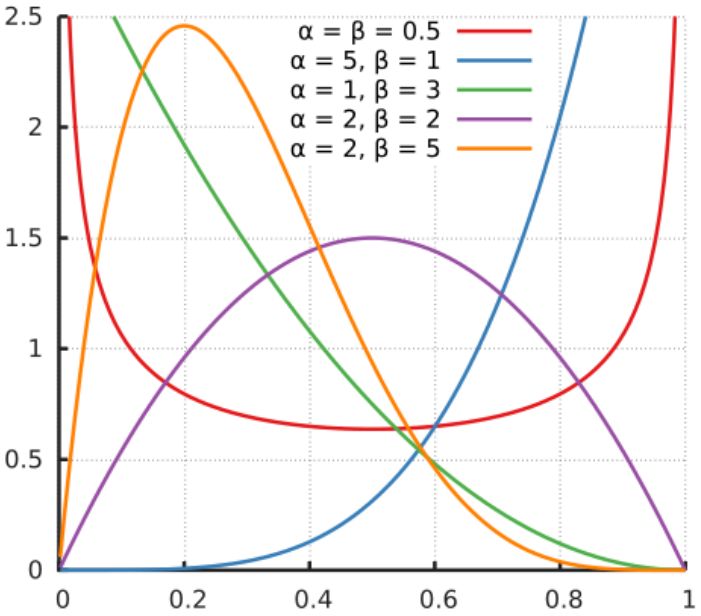

Figura B.1: Función de densidad Beta

Moda

$$\frac{\alpha - 1}{\alpha + \beta - 2}$$

Esperanza

$$E\left[X\right] = \frac{\alpha}{\alpha + \beta}$$

Varianza

$$\operatorname{var}\left(X\right) = \frac{\alpha\beta}{\left(\alpha + \beta\right)^2 \left(\alpha + \beta + 1\right)}$$

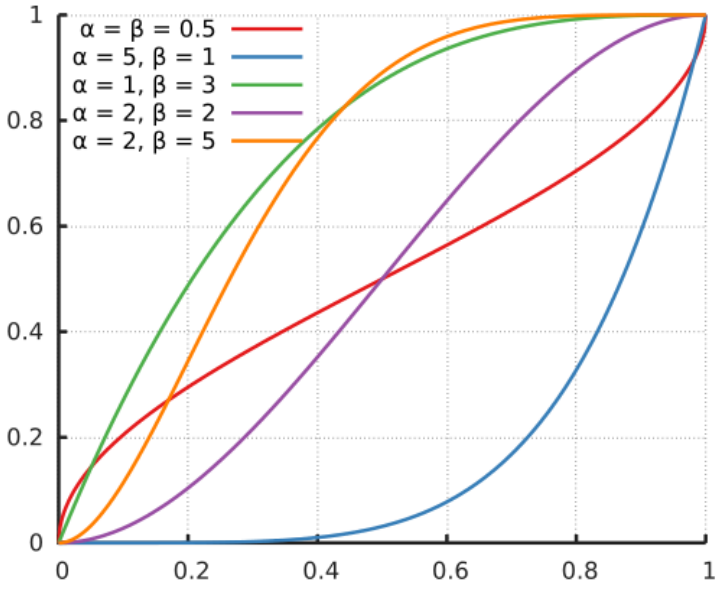

Figura B.2: Función de distribución Beta

Momentos

$$\begin{aligned}
\alpha_k &= E\left[X^k\right] \\
&= \int_0^1 \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha+k-1}(1-x)^{\beta-1}dx \\
&= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \int_0^1 x^{\alpha+k-1}(1-x)^{\beta-1}dx \\
&= \frac{\Gamma(\alpha+k)\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\alpha+k+\beta)}
\end{aligned}$$

Función generatriz de momentos

$$\begin{aligned} M(\theta) &= E\left[e^{\theta X}\right] \\ &= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\int_0^1 e^{\theta x}x^{\alpha-1}(1-x)^{\beta-1} \\ &= \sum_{j=0}^{\infty}\frac{t^j}{j!}E\left[X^j\right] \\ &= \sum_{j=0}^{\infty}\frac{t^j}{\Gamma(j+1)}\frac{\Gamma(\alpha+j)\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\alpha+j+\beta)} \end{aligned}$$

Anexo C

Distribución Dirichlet

En teoría de la probabilidad y estadística, la distribución de Dirichlet, generalmente denotada Dir (α), es una familia de distribuciones de probabilidad continuas multivariada, parametrizadas por un vector α perteneciente al conjunto de los números reales positivos

La función de densidad de la distribución Dirichlet es:

$$f\left(x_1, ..., x_K; \alpha_1, ..., \alpha_K\right) = \frac{1}{B\left(\alpha\right)} \prod_{i=1}^{K} x_i^{\alpha_i - 1}$$

donde

$$x_1, ..., x_K > 0$$

$$x_1 + ... + x_K = 1$$

y B(α) es la función:

$$B\left(\alpha\right) = \frac{\prod_{i=1}^{K} \Gamma\left(\alpha_i\right)}{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)}$$

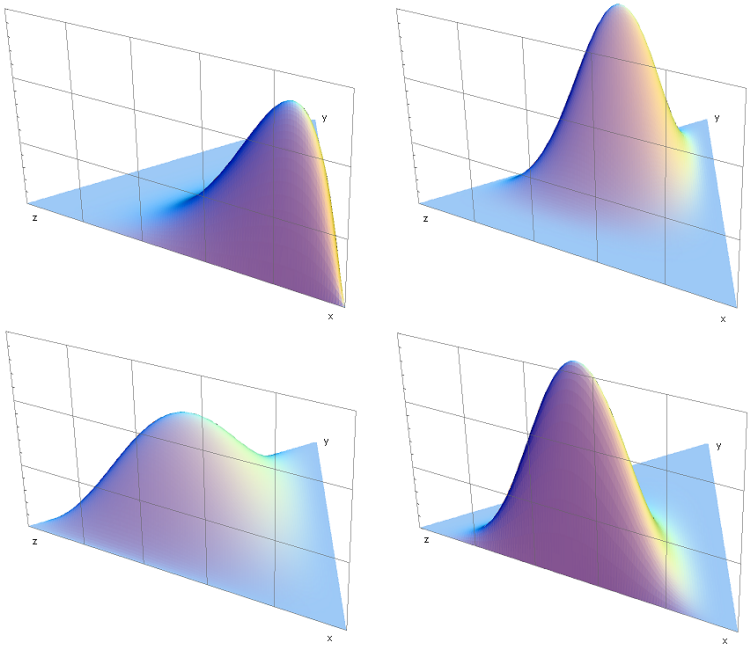

Figura C.1: Función de densidad Dirichlet

Moda

La moda es el vector $(x_1, ..., x_k)$ con

$$x_i = \frac{\alpha_i - 1}{\sum_{i=1}^{K} \alpha_i - K}$$

Esperanza

$$E[X_i] = \frac{\alpha_i}{\sum_{i=1}^{K} \alpha_i}$$

Varianza

$$\operatorname{var}(X_i) = \frac{\alpha_i \sum_{j=1, j\neq i}^{K} \alpha_j}{\left(\sum_{i=1}^{K} \alpha_i\right)^2 \left(\sum_{i=1}^{K} \alpha_i + 1\right)}$$

Covarianza

$$\operatorname{cov}[X_i, X_j] = \frac{-\alpha_i \alpha_j}{\left(\sum_{i=1}^{K} \alpha_i\right)^2 \left(\sum_{i=1}^{K} \alpha_i + 1\right)}$$

Anexo D

Ejemplo de historia de detectives tomado como referencia

La serie Sherlock Yack sirvió, entre otras, como ejemplo para modelar la mente tanto del detective como de los ratones. Debido a su trama simple y modular, decidimos centrarnos en este capítulo para poder aplicarlo directamente. A continuación describimos el capítulo completo formalizando los razonamientos de los personajes.

D.1. Presentación del crimen

El Pelícano canta canciones marineras. Abre la vela de su barco y hay un corazón en vez de su típica calavera. Además le han deshecho la barca y por eso se cae y se hunde.

D.2. Escena del crimen y víctima

Sherlock y Hermione van a ver al Pelícano y la escena del crimen.

Foca se ha ido de vacaciones y ha dado permiso al Pelícano para usar su barco allí.

Hermione se fija en que desataron las cuerdas y el Pelícano dice que primero le hicieron ver lo de la colcha de corazón en vez de su vela. Es decir, pretendían ridiculizarlo, entiende Sherlock.

Por la noche la barca estuvo en medio del lago, así que el culpable puede nadar o volar. También es un maestro bordando, porque es capaz de hacer una colcha muy bien.

Sherlock y Hermione le preguntan al Pelícano por enemigos y admite dos sospechas: Mono Aullador y Señora Panda.

- X (el culpable) ha (humillado y) hundido al Pelícano
- X cambió la vela por una colcha
- La colcha la ha hecho alguien que cose bien $\Longrightarrow$ X podría coser bien
- X llegó a la barca (y la desató)
- Se puede llegar a la barca nadando $\Longrightarrow$ X podría saber nadar
- Se puede llegar a la barca volando $\Longrightarrow$ X podría saber volar
- Pelícano cree que Mono querría humillar a Pelícano $\Longrightarrow$ Pelícano cree que X podría ser Mono
- Pelícano cree que Panda querría humillar a Pelícano $\Longrightarrow$ Pelícano cree que X podría ser Panda

D.3. Visita al Mono

Mono reconoce que le molesta el ruido del Pelícano, cuando graba sus canciones de rock (a pesar de que él hace más ruido todavía).

Descubren por su ropa rota que no sabe coser.

También le preguntan si los heavies saben nadar, a lo que el responde que no.

Pero Sherlock se fija en que hay agua en el suelo y eso le hace ver (explorando el entorno) una boya-patito escondida en el árbol.

El Mono les desafía y les enseña las manos para que le detengan, se ve (si eres observador) que le falta una muñequera.

- Mono admite que querría humillar a Pelícano (habría "móvil")

- Mono tiene la ropa rota
- Mono dice que no sabe coser (y parece factible)
- Mono por definición no puede volar
- Mono dice no saber nadar
- Mono tiene una boya (más adelante se caerá en la cuenta que es otra forma de llegar a la barca)
- A Mono le falta una muñequera

D.4. Visita al Pingüino

Es el pingüino el que les llama denunciando otro crimen.

Se ha producido el robo de la tela de su colcha (segundo misterio anidado).

Descubren que también tiene algo en contra del Pelícano.

Sí sabe nadar... aunque dice que como está teñido no ha podido.

Encuentra una muñequera en el taller del Pingüino, que el Pingüino no dice que sea suya.

Hermione confirma que la colcha es igual que la que usaron con el Pelícano.

- Pingüino dice que hay un W que ha robado una colcha
- Esta colcha, viendo la tela, se ve que es la colcha de la barca (se confirma por observación)
- Pingüino confiesa que querría humillar a Pelícano (habría "móvil")
- Pingüino sabe coser
- Pingüino por definición no puede volar
- Pingüino por definición puede nadar
- Pingüino dice que estos días no puede nadar (pone la excusa del tinte)

- Aparece una muñequera en el taller del Pingüino (más adelante se caerá en la cuenta que es de Mono)
- Pingüino dice que no es suya la muñequera

D.5. Visita a Panda

También confirma que le tiene manía por como canta.

Ven un mantel bordado similar a la colcha y le preguntan si sabe bordar, y efectivamente sabe y le gusta bordar corazones como los de la colcha. También la invitan a la piscina y dice que sabía nadar, pero que como hace mucho que no se mete, duda que sepa.

- Panda admite que querría humillar a Pelícano (habría "móvil").
- Panda sabe coser (y parece factible que hubiese hecho la colcha).
- Panda por definición no puede volar.
- Panda duda cuando le pregunta si sabe nadar (harán un experimento después y se confirmará que no sabe).

D.6. Acusación al culpable

Sherlock hace un resumen: Alguien harto de las canciones del Pelícano, cambió la vela por una colcha que se parece mucho a lo que ha hecho el Pingüino, luego desató la balsa e hizo que se cayera el Pelícano.

Los sospechosos son el Mono, el Pingüino o la Señora Panda.

- X (el culpable) quería humillar al Pelícano.
- Pelícano fue humillado porque su vela fue cambiada por colcha y cayó por tener balsa desatada.
- X cambió la vela por una colcha $\Longrightarrow$ X tenía una colcha $\Longrightarrow$ X llegó a la balsa
- X desató la balsa $\Longrightarrow$ X llegó a la balsa

Tras un rato de reflexión...

Sherlock explica: Panda borda bien pero no sabe nadar.

Panda sabe coser $\Longrightarrow$ Podría haber hecho la colcha.

No se sabe como Panda consiguió llegar a la balsa.

La colcha es de Pingüino pero no puede nadar estos días porque se ha echado el tinte.

Pingüino pregunta si han resuelto su misterio (el del robo de la colcha, segundo misterio anidado) y dicen que sí, precisamente por haber encontrado esa muñequera en su taller.

Pingüino sabe coser. El dice haber hecho la colcha, pero que se la robaron. Asumiendo que eso fuese mentira, pero que lo del tinte fuese verdad ¿cómo llegó a la balsa si ni vuela ni puede nadar estos días aunque sepa? Nos podría estar mintiendo en ambas cosas, pero no es probable porque denunció el robo).

Sherlock dice que el ladrón de la colcha es el mismo que hundió al Pelícano: **Mono**.

X es el Mono

El Mono se defiende diciendo lo de que dijo que él no sabe nadar (uno de los requisitos para poder ser el culpable).

Mono dice que no sabe nadar (ni volar obviamente), con lo que ¿cómo pudo llegar la balsa? (de hecho también podría decir que no sabe bordar asumiendo que él es inocente del robo).

Sherlock explica que lo pudo hacer con la boya-patito. Para ello tuvo primero que robar la colcha del Pingüino (y la prueba que lo hizo es su muñequera perdida). Luego ir con la boya-patito, cambiar la bandera pirata por la colcha y finalmente desatar la balsa.

Mono robó la colcha a Pingüino (lo que se apoya en que la muñequera de Mono estaba en el taller de Pingüino).

Luego Mono llegó hasta la balsa (usando una boya, que por sentido común todos sabemos que es otra forma de ir por el agua... esto se apoya en haber visto la boya en la casa del Mono y en haber visto charcos de agua, demostrando que se usó).

Después Mono cambió la vela por la colcha.

Y finalmente Mono desató balsa.

Ante estos argumentos, el Mono reconoce su culpabilidad.

Anexo E

Ejemplo de ejecución completo

En este apéndice describimos paso a paso una ejecución utilizando los algoritmos desarrollados.

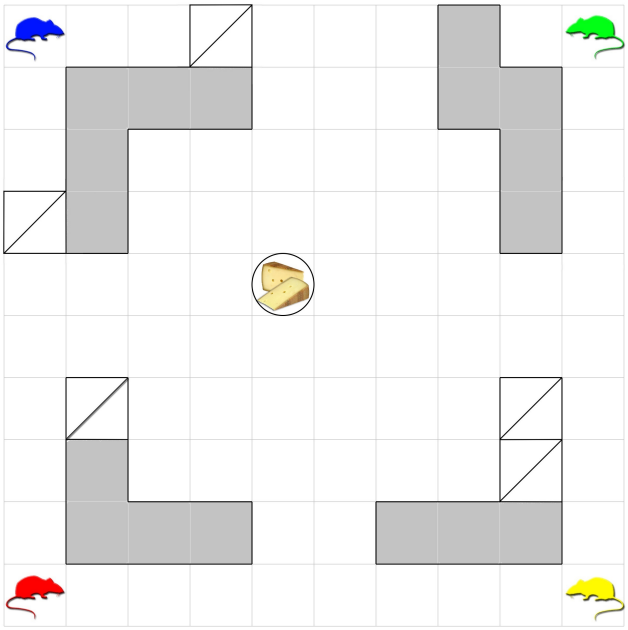

Figura E.1: Estado inicial del tablero utilizado en la ejecución

El laberinto en el que vamos a realizar la ejecución es el que se muestra en la Figura E.1. La partida consta de un total de 20 Turnos. Los ratones utilizan para decidir las acciones una IA inicializada con deseos y para las

preguntas y respuestas la que miente de forma consciente para evitar ser castigado. Los deseos asignados a cada ratón para realizar acciones son los siguientes:

1. Azul
    a) Comerse el queso - 100
    b) No romper shojis - 70
    c) Evitar ser castigado - 10
2. rojo
    a) Caminar - 80
    b) Comerse el queso - 70
    c) No romper shojis - 70
    d) Evitar ser castigado - 10
3. verde
    a) Comerse el queso - 100
    b) No romper shojis - 70
    c) Evitar ser castigado - 10
4. Amarillo
    a) Descansar - 100
    b) No romper shojis - 70
    c) Evitar ser castigado - 10

El flujo de ejecución fue el siguiente:

- Turno: 1
    1. El ratón azul se mueve al este. Posición (1, 2)
    2. El ratón verde se mueve al sur. Posición (2, 10)
    3. El ratón rojo se mueve al este. Posición (10, 2)
    4. El ratón amarillo espera. Posición (10, 10)
- Turno: 2
    1. El ratón azul se mueve al este. Posición (1, 3)

2. El ratón verde se mueve al sur. Posición (3, 10)
3. El ratón rojo se mueve al este. Posición (10, 3)
4. El ratón amarillo espera. Posición (10, 10)

- Turno: 3

1. El ratón azul se mueve al sur. Posición (2, 3)
2. El ratón verde se mueve al sur. Posición (4, 10)
3. El ratón rojo se mueve al este. Posición (10, 4)
4. El ratón amarillo espera. Posición (10, 10)

- Turno: 4

1. El ratón azul se mueve al sur. Posición (3, 3)
2. El ratón verde se mueve al sur. Posición (5, 10)
3. El ratón rojo se mueve al este. Posición (10, 5)
4. El ratón amarillo espera. Posición (10, 10)

- Turno: 5

1. El ratón azul se mueve al este. Posición (3, 4)
2. El ratón verde se mueve al oeste. Posición (5, 9)
3. El ratón rojo se mueve al norte. Posición (9, 5)
4. El ratón amarillo espera. Posición (10, 10)

- Turno: 6

1. El ratón azul se mueve al este. Posición (3, 5)
2. El ratón verde se mueve al oeste. Posición (5, 8)
3. El ratón rojo se mueve al norte. Posición (8, 5)
4. El ratón amarillo espera. Posición (10, 10)

- Turno: 7

1. El ratón azul se mueve al sur. Posición (4, 5)
2. El ratón verde se mueve al oeste. Posición (5, 7)
3. El ratón rojo se mueve al norte. Posición (7, 5)
4. El ratón amarillo espera. Posición (10, 10)

- Turno: 8
    1. El ratón azul se mueve al sur. Posición (5, 5)
    2. El ratón verde se mueve al oeste. Posición (5, 6)
    3. El ratón rojo se mueve al norte. Posición (6, 5)
    4. El ratón amarillo espera. Posición (10, 10)
- Turno: 9
    1. El ratón azul come. Posición (5, 5)
    2. El ratón verde se mueve al este. Posición (5, 7)
    3. El ratón rojo espera. Posición (6, 5)
    4. El ratón amarillo espera. Posición (10, 10)
- Turno: 10
    1. El ratón azul espera. Posición (5, 5)
    2. El ratón verde se mueve al este. Posición (5, 8)
    3. El ratón rojo espera. Posición (6, 5)
    4. El ratón amarillo espera. Posición (10, 10)
- Turno: 11
    1. El ratón azul espera. Posición (5, 5)
    2. El ratón verde se mueve al este. Posición (5, 9)
    3. El ratón rojo espera. Posición (6, 5)
    4. El ratón amarillo espera. Posición (10, 10)
- Turno: 12
    1. El ratón azul espera. Posición (5, 5)
    2. El ratón verde se mueve al este. Posición (5, 10)
    3. El ratón rojo se mueve al sur. Posición (7, 5)
    4. El ratón amarillo espera. Posición (10, 10)
- Turno: 13
    1. El ratón azul se mueve al norte. Posición (4, 5)
    2. El ratón verde se mueve al sur. Posición (6, 10)

3. El ratón rojo se mueve al sur. Posición (8, 5)
4. El ratón amarillo espera. Posición (10, 10)

- Turno: 14

1. El ratón azul se mueve al oeste. Posición (4, 4)
2. El ratón verde se mueve al sur. Posición (7, 10)
3. El ratón rojo se mueve al sur. Posición (9, 5)
4. El ratón amarillo espera. Posición (10, 10)

- Turno: 15

1. El ratón azul se mueve al norte. Posición (3, 4)
2. El ratón verde se mueve al sur. Posición (8, 10)
3. El ratón rojo se mueve al sur. Posición (10, 5)
4. El ratón amarillo espera. Posición (10, 10)

- Turno: 16

1. El ratón azul se mueve al oeste. Posición (3, 3)
2. El ratón verde se mueve al sur. Posición (9, 10)
3. El ratón rojo se mueve al oeste. Posición (10, 4)
4. El ratón amarillo espera. Posición (10, 10)

- Turno: 17

1. El ratón azul se mueve al oeste. Posición (3, 2)
2. El ratón verde se mueve al sur. Posición (10, 10)
3. El ratón rojo se mueve al oeste. Posición (10, 3)
4. El ratón amarillo espera. Posición (10, 10)

- Turno: 18

1. El ratón azul se mueve al oeste. Posición (3, 1)
2. El ratón verde se mueve al norte. Posición (9, 10)
3. El ratón rojo se mueve al oeste. Posición (10, 2)
4. El ratón amarillo espera. Posición (10, 10)

- Turno: 19

1. El ratón azul se mueve al norte. Posición (2, 1)
2. El ratón verde se mueve al norte. Posición (8, 10)
3. El ratón rojo se mueve al oeste. Posición (10, 1)
4. El ratón amarillo espera. Posición (10, 10)

- Turno: 20

1. El ratón azul se mueve al norte. Posición (1, 1)
2. El ratón verde se mueve al norte. Posición (7, 10)
3. El ratón rojo espera. Posición (10, 1)
4. El ratón amarillo espera. Posición (10, 10)

Los ratones acaban por lo tanto en las siguientes posiciones:

Color	**Posición**
Azul	(1,1)
Verde	(7, 10)
Rojo	(10, 1)
Amarillo	(10,10)

Una vez acabada la ejecución el científico se despierta, observa que el queso no está y comienza el interrogatorio para atrapar al culpable. El flujo de esta fase es el siguiente:

- Ratón azul, ¿te has comido el queso?. Su respuesta es No.
- Ratón verde, ¿te has comido el queso?. Su respuesta es No.
- Ratón rojo, ¿te has comido el queso?. Su respuesta es No.
- Ratón amarillo, ¿te has comido el queso?. Su respuesta es No.

Nadie se delata por lo que a priori no tenemos ningún culpable. Cuando acaba de preguntar por el queso comienza a preguntar por los shojis rotos. En esta ejecución el único shoji que se rompe es el (1,3) por lo que solo preguntará por este shoji:

- Ratón azul, ¿has estado en la casilla (1,3)? Su respuesta es No.
- Ratón verde, ¿has estado en la casilla (1,3)? Su respuesta es No.

- Ratón rojo, ¿has estado en la casilla (1,3)? Su respuesta es No.
- Ratón amarillo, ¿has estado en la casilla (1,3)? Su respuesta es No.

Como ninguno dice haber estado allí, culpa al ratón azul porque es el que más cerca tiene su casilla inicial del shoji roto. Además está en su camino más cercano. Una vez tiene un sospechoso, decide preguntar a los ratones si dicho sospechoso lo consideran culpable o no:

- Ratón azul, ¿se ha comido el ratón amarillo el queso?. Su respuesta es No lo sé.
- Ratón verde, ¿se ha comido el ratón azul el queso?. Su respuesta es Sí.
- Ratón rojo, ¿se ha comido el ratón azul el queso?. Su respuesta es Sí.
- Ratón amarillo, ¿se ha comido el ratón azul el queso?. Su respuesta es Sí.

Una vez que corrobora que el ratón sospechoso se ha comido el queso emite su juicio final:

Color	**Culpabilidad**
Azul	45.5 %
Verde	32.5 %
Rojo	35.75 %
Amarillo	35.75 %

Por lo tanto el castigado es el ratón azul.

Referencias

[1] Computer passes Turing test by imitating 13-year-old boy, 'Eugene Goostman'. *The Huffington Post*, 2014.

[2] Nahum Alvarez and Federico Peinado. Modelling suspicion as a game mechanism for designing a computer-played investigation character.

[3] Michel Amelin. Sherlock Yack. illustrated by Ruth Christelle, 2011. Mondo TV.

[4] R Bosley. Handbook of the history of logic, vol 1, Greek, Indian and Arabic logic., 2005.

[5] Michael Bratman. Intention, plans, and practical reason. 1987.

[6] Carlos Carrascosa, Javier Bajo, Vicente Julián, Juan M Corchado, and V Botti. Hybrid multi-agent architecture as a real-time problem-solving model. *Expert Systems with Applications*, 34(1):2–17, 2008.

[7] Valve Corporation. Half-life. Microsoft Windows, 1998.

[8] Valve Corporation. Counter-Strike. Microsoft Windows, 1999.

[9] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. Social coding in GitHub: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1277–1286. ACM, 2012.

[10] Mark d'Inverno, Michael Luck, Michael Georgeff, David Kinny, and Michael Wooldridge. The dMARS architecture: A specification of the distributed multi-agent reasoning system. *Autonomous Agents and Multi-Agent Systems*, 9(1-2):5–53, 2004.

[11] Naughty Dog. Crash Bandicoot. PlayStation, 1996.

[12] Naughty Dog. Uncharted 4: A thief's end. PlayStation 4, 2016.

[13] Howard Eves. An introduction to the history of Mathematics. Technical report, 1990.

[14] Richard E Fikes and Nils J Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[15] Game Freak. Pokémon Rojo y Azul. Game Boy, 1996.

[16] Carlos Fresneda. Un ordenador logra superar por primera vez el test de Turing. *El Mundo*, 2014.

[17] Future Games. The Black Mirror. Microsoft Windows, 2003.

[18] Michael P Georgeff and Amy L Lansky. Procedural knowledge. *Proceedings of the IEEE*, 74(10):1383–1398, 1986.

[19] James Gosling. *The Java language specification*. Addison-Wesley Professional, 2000.

[20] JS Gu and CW De Silva. Development and implementation of a real-time open-architecture control system for industrial robot systems. *Engineering Applications of Artificial Intelligence*, 17(5):469–483, 2004.

[21] E Friedman Hill. Jess in action: rule-based systems in Java, 2003.

[22] Audun Jøsang. Artificial reasoning with subjective logic. In *Proceedings of the second Australian workshop on commonsense reasoning*, volume 48, page 34. Citeseer, 1997.

[23] Audun Jøsang. A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(03):279–311, 2001.

[24] Audun Josang. Conditional reasoning with subjective logic. *Journal of Multiple-Valued Logic and Soft Computing*, 15(1):5–38, 2008.

[25] Victor J Katz. A history of mathematics: an introduction. *Addison-Wesley*, 1998.

[26] Richard Levinson & William Link. Columbo, 1971.

[27] Monolith. Shogo. Microsoft Windows, Linux, Mac OS, 1998.

[28] James Moor. *The Turing test: the elusive standard of artificial intelligence*, volume 30. Springer Science & Business Media, 2003.

[29] Jeff Orkin. Three states and a plan: the AI of FEAR. In *Game Developers Conference*, volume 2006, page 4, 2006.

[30] Jeff Orkin-Monolith Productions. Applying Goal-Oriented Action Planning to games.

[31] Monolith Productions. The Operative: No One Lives Forever. PlayStation 2, Microsoft Windows, Mac OS X, 2000.

[32] Monolith Productions. No One Lives Forever 2: A Spy in H.A.R.M.'s Way. Microsoft Windows, Mac OS X, 2002.

[33] Monolith Productions. F.E.A.R. First Encounter Assault Recon. PlayStation 3, Xbox 360, Microsoft Windows, 2005.

[34] Sucker Punch Productions. Sly Cooper and the Thievius Raccoonus. PlayStation 2, 2002.

[35] Anand S Rao, Michael P Georgeff, et al. BDI agents: From theory to practice. In *ICMAS*, volume 95, pages 312–319, 1995.

[36] James Owen Ryan, Adam Summerville, Michael Mateas, and Noah Wardrip-Fruin. Toward characters who observe, tell, misremember, and lie. *Proc. Experimental AI in Games*, 2, 2015.

[37] Team Silent. Silent Hill. PlayStation, 1999.

[38] Rocksteady Studios. Batman: Arkham Knight. PlayStation 4, Xbox One, 2015.

[39] Tecmo. Project Zero. PlayStation 2, 2001.

[40] Christof Teuscher. *Alan Turing: Life and legacy of a great thinker*. Springer Science & Business Media, 2004.

[41] Alan M Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[42] Ubisoft. Assassin's Creed III: Liberation. PlayStation 3, Xbox 360, 2012.

[43] Konstantin Vikhorev, Natasha Alechina, and Brian Logan. The ARTS real-time agent architecture. In *International Workshop on Languages, Methodologies and Development Tools for Multi-Agent Systems*, pages 1–15. Springer, 2009.

[44] Michael J Wooldridge. *Reasoning about rational agents*. MIT press, 2000.